PEER-TO-PEER STREAMING: DESIGN AND CHALLENGES

by

NAZANIN MAGHAREI

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

December 2010

"Peer-to-Peer Streaming: Design and Challenges," a dissertation prepared by Nazanin Magharei in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science. This dissertation has been approved and accepted by:

Prof. Reza Rejaie, Chair of the Examining Committee

Date

Committee in charge:

Prof. Reza Rejaie, Chair
Prof. Virginia Lo
Prof. Jun Li
Prof. David Levin
Prof. Markus Hofmann

Accepted by:

Dean of the Graduate School

An Abstract of the Dissertation of

Nazanin Magharei for the degree of Doctor of Philosophy

in the Department of Computer and Information Science

to be taken December 2010

Title: PEER-TO-PEER STREAMING: DESIGN AND CHALLENGES

Approved: ______________________________

Prof. Reza Rejaie

Streaming multimedia content over the Internet is extremely popular mainly due to emerging applications such as IPTV, YouTube and e-learning. All these applications require simultaneous streaming of multimedia content from one or multiple sources to a large number of users. Such applications impose unique requirements in terms of server bandwidth and playback delay which are difficult to achieve in a scalable fashion with the traditional client-server architecture. Peer-to-peer (P2P) overlays offer a promising approach to support scalable streaming applications, that we broadly refer to as “P2P streaming”. Design of a scalable P2P streaming mechanism that accommodates heterogeneity of peers' bandwidth and copes with dynamics of peer participation while ensuring in-time delivery of the multimedia content to individual peers is extremely challenging. Besides these fundamental challenges, P2P streaming applications are facing practical issues such as encouraging peers' contribution and decreasing the costly inter-ISP P2P traffic.

In this dissertation, we study several aspects of live P2P streaming with the goal of improving the performance of such systems. This dissertation can be categorized into two parts as follows. *(i)* We present the design and evaluation of a mesh-based live P2P streaming mechanism, called PRIME. Further, we perform a head-to-head comparison between the two approaches on live P2P streaming, namely tree-based and mesh-based. We demonstrate the superiority of the mesh-based approach. In the quest for a systematic comparison of existing mesh-based solutions on live P2P streaming, we leverage the insights from our design in PRIME and propose an evaluation methodology. Utilizing the evaluation methodology, we compare the performance of existing mesh-based live P2P streaming solutions. *(ii)* From a more practical perspective, we tackle some of the existing practical issues in the deployment of live P2P streaming applications, namely providing incentives for participating peers to contribute their resources and designing ISP-friendly live P2P streaming protocols with the ultimate goal of reducing costly inter-ISP traffic. In the end, this dissertation reveals fundamental trade-offs in the design, comparison and meaningful evaluation of basic and practical live P2P streaming mechanisms under realistic settings.

This dissertation includes my previously published and my co-authored materials.

CURRICULUM VITAE

NAME OF AUTHOR: Nazanin Magharei

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, USA
Sharif University of Technology, Tehran, Iran

DEGREES AWARDED:

Doctor of Philosophy in Computer and Information Science,
2010, University of Oregon

Bachelor of Science in Electrical Engineering,
2002, Sharif University of Technology

AREAS OF SPECIAL INTEREST:

Computer Networks
Peer-to-Peer Overlays
Multimedia Streaming

PROFESSIONAL EXPERIENCE:

Graduate Research Fellow, Department of Computer and Information Science, University of Oregon, 2004 - present

Summer Intern, Bell Labs, Alcatel-Lucent Inc., Murray Hill, NJ, 6/2008 - 8/2008

Summer Intern, Thomson Research Lab, Princeton, NJ, 6/2006 -1/2007

Network Engineer, Computing Center, Sharif University of Technology, Tehran, Iran, 10/1998-8/2004

AWARDS AND HONORS:

Marthe E. Smith Memorial Science Fellowship, University of Oregon, 2009-2010.

Henry V. Howe Scholarship, University of Oregon, 2008-2009

Miller Family Graduate Award, University of Oregon, 2008-2009

Juilf Scholarship Award, University of Oregon, 2006-2007

Student Travel Grant, ICNP 2005

PUBLICATIONS:

A. Rasti, N. Magharei, R. Rejaie, and W. Willinger. “Eyeball ASes: From Geography to Connectivity,” *Proc. ACM/USENIX Internet Measurement Conference*, 2010.

N. Magharei, R. Rejaie, and Y. Guo, “Incorporating contribution-awareness into mesh-based peer-to-peer streaming services,” *Peer-to-Peer Networking and Applications*, 2010.

N. Magharei and R. Rejaie, “ISP-friendly P2P streaming,” *IEEE MULTIMEDIA COMMUNICATIONS TECHNICAL COMMITTEE E-Letter*, 2009.

N. Magharei and R. Rejaie, “PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming,” *IEEE/ACM Transactions on Networking*, 2009.

N. Magharei and R. Rejaie, “Overlay monitoring and repair in swarm-based peer-to-peer streaming,” In *Proc. ACM NOSSDAV*, 2009.

N. Magharei, Y. Guo, and R. Rejaie, “Issues in Offering Live P2P Streaming Service to Residential Users,” In *Proc. IEEE CCNC*, 2007.

N. Magharei and R. Rejaie, “Prime: Peer-to-peer receiver-driven mesh-based streaming,” In *Proc. IEEE INFOCOM*, 2007.

N. Magharei, R. Rejaie, and Y. Guo, “Mesh or Multiple-Tree: A Comparative Study of P2P Live Streaming Services,” In *Proc. IEEE INFOCOM*, 2007.

N. Magharei and R. Rejaie, “Understanding Mesh-based Peer-to-Peer Streaming,” In *Proc. ACM NOSSDAV*, 2006.

N. Magharei and R. Rejaie, “Adaptive Receiver-Driven Streaming from Multiple Senders,” *ACM/SPIE Multimedia Systems Journal*, 2006.

N. Magharei, A. H. Rasti, D. Stutzbach, and R. Rejaie, “Peer-to-peer receiver-driven mesh-based streaming,” In *Proc. ACM SIGCOMM, Poster Session*, 2005.

ACKNOWLEDGMENTS

First, I would like to thank my parents, Fahimeh and Hassan for their unconditional love, sacrifices and support throughout my life. They have instilled in me the passion of learning, inspired me to always strive for excellence and encouraged me to be a strong thinker. I am eternally grateful to them for believing in me and for all the opportunities they made available to me. Especially, I would like to thank my beloved husband Amir, whose emotional support, tolerance and help allowed me to pursue this degree. Thanks for standing by me through ups and downs, bearing with me in every step along the way and the colorful life you have given me.

I would like to express my gratitude to my advisor, Prof. Reza Rejaie, who made this dissertation come into being by his invaluable guidance and insightful advices. His strict training, leadership and openness for discussions have greatly helped me finish my dissertation. I appreciate his tremendous efforts not only on my academic research, but also on leadership skills, interpersonal skills and career choices. I would also like to thank my committee members, Prof. Virginia Lo, Prof. Jun Li, Prof. David Levin, and Prof. Markus Hofmann for taking the time to review my dissertation and giving valuable suggestions and feedback.

My thanks also go out to Dr. Yang Guo, Dr. Volker Hilt and Dr. Ivica Rimac for insightful comments and collaborative work.

I owe special thanks to my friends and colleagues in the Mirage research lab, Mojtaba Torkjazi and Masoud Valafar for their invaluable support and suggestions throughout my PhD study. They were wonderful group-mates and friends and make my last three years of study an enjoyable and memorable journey.

In closing, I would like to dedicate this dissertation to the memory of a very special person in my life, my grandfather, Jalal Magharei, who continually encouraged and motivated me to pursue academic excellence. He believed that acquiring knowledge and wisdom broaden one's life perspectives. I truly wish he was with me today to share in this achievement, but I am sure he is proud of my accomplishments.

To my parents, Fahimeh and Hassan,

and to my husband, Amir

This dissertation is specially dedicated to Jalal Magharei

whose memory always stays in my heart

TABLE OF CONTENTS

Chapter Page

Chapter Page

Chapter Page

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

Streaming of multimedia content (*i.e.*, video and audio) over the Internet is extremely popular as witnessed by emerging applications such as IPTV, YouTube and e-learning. Multimedia streaming has the potential to enrich peoples' lives and create new business opportunities.

Streaming applications require simultaneous delivery of multimedia content from one or multiple servers to a large number of users. Such applications impose unique requirements in terms of server bandwidth and playback delay to application designers. The size of a typical video content is an order of magnitude larger than other types of content. Moreover, the streaming content should be delivered in real time to all users to maintain smooth playback at the streaming rate. Such requirements limit the number of users that a video server can support simultaneously. Thus, due to the limited scalability of the traditional client-server content delivery architecture, the design of a scalable architecture for delivery of multimedia streaming has been widely motivated.

As broadband Internet access has become available to many users, a new content delivery architecture known as peer-to-peer (P2P) has emerged to replace traditional client-server architecture for delivery of multimedia streaming. A P2P infrastructure does not rely on any special support from the core of the network as required by IP multicast. Instead, participating users (called peers) form an overlay and contribute their bandwidth to delivery of the content among themselves at the edge of the network. In essence each peer can act as a client, which receives the content as well as a server, that forwards the content to other participating peers. P2P has been leveraged in different contexts, namely, file sharing, stored multimedia streaming and live multimedia streaming. In this dissertation, we focus on live P2P streaming systems.

A typical P2P streaming session might have a range of sizes from 100 to 1M peers depending on the popularity of the streaming content. In a P2P streaming session, streaming content is broadcast by a single source or multiple sources concurrently to interested peers. Such source(s) can be average Internet users that host a TV-show from home or broadcast a live sporting event. Peers, which are ordinary Internet users, join and leave their desired streaming session at any time. Such peers have heterogeneous capabilities in terms of processing power and availability or willingness to contribute bandwidth. Throughout this dissertation, we consider live P2P streaming over a single streaming session that is broadcast by a single source and supports a large number of interested peers.

1.1. Challenges in Live P2P Streaming

The growing interest in P2P for multimedia streaming is due to two major reasons: (i) This architecture does not require any special support from the network, which leads to ease of deployment and low cost. (ii) P2P is self-scaling, as the total resources (*e.g.*, bandwidth or CPU) organically grows with the number of participating peers. Without loss of generality, in this dissertation, we consider live P2P streaming in a single streaming channel that supports a large number of interested peers. Despite the scalability of P2P compared to client-server architecture, P2P introduces a set of new challenges for delivery of multimedia streaming to a significantly large number of interested and dynamic users. The most relevant challenges that are driving the research activity in the P2P streaming context are as follows:

Scalability -Bandwidth of individual peers can be limited, heterogeneous, and asymmetric and it can change over time. Therefore, while resources may scale organically with the number of peers, utilization of these limited dynamic resources to support a smooth playback at a constant streaming rate to all peers is crucial in the scalability of P2P streaming.

Resiliency -In P2P systems, peers may join and leave the overlay at any time. An existing connection between two peers may be disconnected and a new connection can be established at any time. In fact, due to uncertainty in peers' behavior, P2P systems are inherently dynamic and unreliable. These dynamics and unreliabilities can introduce obstacles in maintaining a sustainable and smooth streaming rate to all participating peers.

Contribution Awareness & Fairness - Practical deployment of live P2P streaming applications introduces a whole set of issues. The self scalability of P2P systems becomes violated when one relies completely on ordinary peers for streaming as peers may intentionally or unintentionally contribute less than their demand of resources. In the extreme cases, peers may be ”free-loaders” by consuming bandwidth without contributing any. How to encourage or incentivize peers to contribute their resources and how to allocate the available resources in a fair manner is challenging.

Traffic Locality -Another practical issue for P2P streaming applications is the huge amount of network traffic that imposes a high cost for ISPs. This problem arises from the fact that P2P applications usually build network agnostic overlays which can be very inefficient in using network resources. In fact, P2P applications can result in redundant traffic between ISPs which results in a high cost for the corresponding ISPs.

Security -As P2P systems rely on unknown Internet users, many security concerns arise. Content integrity, robustness to denial of service, malicious user behavior (*i.e.*, manipulation of protocols, Sybil attacks, etc.) are among the few security concerns in P2P systems.

To address these challenges, P2P streaming protocols should maximize the utilization of the available resources (*i.e.*, server and peers' bandwidth), accommodate scalability and sudden changes in the peer population, cope with heterogeneous peer bandwidth and dynamics in peer participation and be resilient to network variations. Moreover, to deploy P2P streaming protocols in practice, fair distribution of resources among peers, ISP-friendliness and security concerns must be addressed.

1.2. Dissertation Scope & Contributions

In this dissertation, we study several aspects of mesh-based live P2P streaming with the ultimate goal of improving the performance of live P2P streaming over the Internet. Towards this goal, we tackle a subset of research problems in the area of live P2P streaming. This dissertation can be categorized into two broad parts as follows: *(i)* We designed and evaluated PRIME, a novel mesh-based approach to live multimedia streaming. Further, we examined the similarities and differences between mesh-based and tree-based approaches for live P2P streaming and performed a head-to-head comparison between these two approaches. In the quest for a systematic comparison of existing mesh-based solutions on live P2P streaming, we developed an evaluation methodology that assists us in performing such a comparison. *(ii)* From a more practical perspective, we tackled some of the important issues in the practical deployment of live P2P streaming applications, namely, providing incentives for participating peers to actively contribute their resources and designing an ISP-friendly live P2P streaming protocol with the ultimate goal of reducing costly inter-ISP traffic. Below, we give an overview of the dissertation contributions.

1.2.1. Design of a Mesh-based Live P2P Streaming

Through a performance-driven approach, we design and evaluate PRIME, a scalable mesh-based P2P streaming mechanism for live content. The main design goal of PRIME is to minimize two performance bottlenecks, namely the bandwidth bottleneck and the content bottleneck. We show that the global pattern of content delivery should consist of two phases of diffusion followed by swarming. This leads to effective utilization of available resources to accommodate scalability and also minimizes the content bottleneck.

Through extensive packet level simulations, we carefully examine the impact of overlay connectivity, block scheduling scheme at individual peers, and source behavior on the overall performance of PRIME. Our results reveal fundamental design tradeoffs for mesh-based P2P streaming of live content.

1.2.2. Performance Comparison of Existing Approaches on Live P2P Streaming

Existing approaches for P2P streaming can be divided into two general categories: *(i) tree-based* that uses push-based content delivery over multiple tree-shaped overlays and *(ii) mesh-based* that uses swarming content delivery over a randomly connected mesh. Previous studies focused only on a particular P2P streaming mechanism and thus, no comparison between these two categories has been conducted. This dissertation compares and contrasts the performance of representative protocols of tree-based and mesh-based approaches. Towards that, we identify some striking similarities and differences between these two approaches.

Through both packet-level and session-level simulations, we separately examine the behavior of content delivery and overlay construction mechanisms for both approaches in static and dynamic scenarios. The results indicate that the mesh-based approach consistently exhibits a superior performance over the tree-based approach. This dissertation also shows the main factors causing the inferior performance of the tree-based approach.

1.2.3. Evaluation Methodology for the Comparison of Mesh-based Live P2P Streaming Solutions

There are many proposed mechanisms on mesh-based solutions for live P2P. The performance of these mechanisms are often presented in terms of delivered quality to individual peers in a specific setting without demonstrating the underlying performance bottlenecks. However, the performance of mesh-based live P2P streaming mechanisms depends on the overall effects of several factors, namely, connectivity of the overlay, block scheduling scheme and environment settings. With the goal of systematic comparison of mesh-based mechanisms, we present an evaluation methodology for mesh-based live P2P streaming mechanisms. Our methodology leverages the insights from our design of PRIME and presents a set of metrics to capture the behavior of mesh-based P2P streaming mechanisms. In particular, we present a set of signatures to easily identify good behavior in mesh-based P2P streaming mechanisms.

We cover the spectrum of solutions for mesh-based live P2P streaming and demonstrate the effectiveness of our evaluation methodology. We identify the performance bottlenecks of each solution in both resource-rich and poor scenarios. This comparison provides useful insights in the design of live P2P mesh-based mechanism and our methodology offers a unified framework for head-to-head comparison of various mesh-based solutions.

1.2.4. Resources Management in Live P2P Streaming

As we have discussed, the feasibility of scalable P2P streaming depends on the availability of resources, *i.e.*, outgoing bandwidth, proportional to the number of participating peers. However, in practice the contributed resources are often insufficient as a subset of peers are unable or unwilling to contribute as much bandwidth as they demand. In a randomly connected P2P overlay, the impact of decrease in resources on the delivered quality to peers is not correlated to the peers' contribution. As the allocated bandwidth to each peer determines its delivered quality, a reasonable approach in such a *resource-constrained* setting is to allocate resources to individual peers proportional to their contribution. In the dissertation, we propose a tax-based contribution-aware mechanism that provides incentives for participating peers to actively contribute their resources in order to improve their delivered quality in the context of mesh-based live P2P streaming. The contribution-aware mechanism accommodates the heterogeneity, asymmetry and dynamic nature of available resources.

Focusing on the connectivity of individual peers as first-degree approximation of their allocated resources, we use a session-level simulator to investigate the effect of key design parameters (*e.g.*, tax rate and preemption policies) over a wide range of scenarios using a realistic model for peer dynamics and pairwise delay. In particular, we examine the effect of aggregate available resources, distribution of contributed resources among peers, and group size on the allocation and overall utilization of resources, as well as the stability of the overlay.

1.2.5. P2P Traffic Localization in Live Streaming

The popularity and high bandwidth demand of mesh-based P2P streaming applications potentially can result in generating a significant amount of network traffic. Because building a random overlay among peers ignores the underlying network topology, the generated traffic imposes a high cost on ISPs. Localization or limiting inter-ISP P2P traffic, can potentially affect the performance of mesh-based live P2P streaming applications. In this dissertation, we investigate the impact of localization on the performance of mesh-based live P2P streaming applications and show that by increasing the localization the performance degrades significantly. To achieve ISP-friendliness without compromising the delivered quality to peers, we adopt two orthogonal approaches: *(i)* revising overlay connectivity scheme and *(ii)* revising the block scheduling scheme.

In the first approach, we change the overlay connectivity by allowing peers to probabilistically react to a shortage of content and establish proper connections between various regions in a distributed fashion without any coordination among peers. We present a distributed overlay monitoring and repair mechanism that maintains proper connectivity of the overlay and minimizes the inter-ISP traffic while ensuring high quality stream to peers. We examine the effectiveness of our proposed mechanism and show how rapidly it improves the connectivity of the overlay with a minimum amount of rewiring.

The second approach revises the block scheduling scheme. We propose a novel two-tier overlay-aware block scheduling scheme that maximizes the traffic localization while delivering high quality stream to individual peers. Through extensive simulations, we demonstrate the ability of this approach to deliver a high quality stream over a fully localized overlay in various realistic scenarios.

1.3. Dissertation Outline

The dissertation is organized as follows. We present related work and background on this research in Chapter II. Chapter III presents our design and evaluation of a mesh-based P2P live streaming mechanism. In Chapter IV, we discuss the similarities and differences between existing approaches on live P2P streaming and compare their performance. We propose our framework on evaluation of live P2P mesh-based streaming mechanisms in Chapter V, and further systematically compare the existing mesh-based solutions in various settings. Chapter VI approaches the practical issue of resource management and incentives in the context of live P2P streaming systems. Chapter VIII and VII present our work on the issue of P2P traffic localization in the context of mesh-based live P2P streaming. Finally, we present concluding remarks, summary of contributions and future directions in Chapter IX.

Inclusion of Published Material: All chapters of this dissertation are based heavily on my published or intended to be published papers with co-authors [1, 2, 3, 4, 5, 6, 7, 8, 9]. In all of the work, the experimental work is entirely mine, with my co-authors contributing technical guidance, editorial assistance, and small portions of writing.

CHAPTER II

RELATED WORKS

In this chapter, first, we present existing studies on the design of P2P streaming applications and categorize the existing solutions according to several aspects. Towards that, we discuss some background on various types of streaming content and further, focus on existing approaches on live P2P streaming. Second, we discuss relevant studies on some of the practical challenges that today's P2P live streaming applications are facing which are in the scope of this dissertation.

2.1. Design of P2P Streaming: Types of Streaming Content

During the past decade, the following three major architectures have been used to support delivery of streaming content over the Internet: *(i) unicast or client-server*, *(ii) IP multicast*, and *(iii) peer-to-peer*. The traditional unicast or client-server architecture allows a single server to exchange content to clients in a one-to-one communication model. However, this architecture does not scale with the number of clients due to the limited server's resources (*i.e.*, bandwidth and CPU). Many streaming applications require one-to-many or many-to-many communication model. However, multiple unicasts is not scalable and efficient for that purpose which motivates IP multicast architecture. By relying on the core of the network, IP multicast allows delivery of a single copy of the streaming content from a single server to any number clients in a convenient and efficient fashion. The limited availability of IP multicast due to its deployment difficulties has motivated a new architecture called peer-to-peer (P2P) which pushes the delivery functionality to the edge of the network. This architecture has received a great deal of attention for support of delivery of streaming content over the Internet.

In P2P, participating peers form a P2P overlay and forward the content through the overlay links. Each peer acts as a client, which receives the content as well as a server, that forwards the content to other participating peers. The growing interest in P2P for streaming is due to two major reasons: *(i)* This architecture does not require any special support from the network, which leads to its ease of deployment and low cost. *(ii)* P2P is self-scaling, as the total resources (*e.g.*, bandwidth, CPU) organically increase with the number of participating end-systems.

In P2P streaming applications, the content is streamed from a single or multiple sources to the receiver(s). Receiver can start playing as soon as a few packets are received. Stream has an inherent playback rate (*e.g.*, MPEG 1.4 Mbps) which depends on the coding rate. Streaming content delivery has *timing constraint*, that is once playback started/initiated, packets should be delivered before their playtime, otherwise an interruption will happen. Streaming content can be categorized into *On-demand* and *live streaming* [10].

2.1.1. On-demand

On-demand streaming is referred to streaming a pre-stored/cached audio or video from a source to interested receivers [11, 12, 13, 14, 15]. In on-demand streaming, the entire content is available and stored which gives control to both the source and receivers. Source can deliver the stream at a higher rate and receivers can choose the playtime (*i.e.*, VCR functionality which is fast forward, pause and rewind).

In P2P on-demand streaming applications, receiver can delay its playback and starts buffering to absorb any variation of bandwidth and minimize probability of late arrival of packets. This flexibility results in a *loose timing constraint* for such streaming applications.

2.1.2. Live

In live P2P streaming, content is generated on the fly and can be delivered as soon as it becomes available. Therefore, the average sending rate is equal to the stream rate and source cannot send faster. Moreover, advancing playtime is not feasible in this context, as the future content is not available. Live streaming can be non-interactive or interactive.

Live Non-interactive Live non-interactive streaming refers to the synchronized distribution of live content from a single source to receivers. An example is broadcasting a live sporting event. In this context, a receiver can delay its playback and starts buffering. However, the delay should be small due to the live nature of the content. Therefore, live non-interactive streaming has a *relatively* loose timing constraint.

Live Interactive Streaming Live interactive streaming involves a group of users who use audio/video to communicate with each other in real time. Essentially, users interact with each other by exchanging timely data that is generated on the fly.Some examples of live interactive streaming are IP telephony, video conferencing and distributed interactive gaming applications. In this context, the distinction between the role of source and receivers is diminished as each user is able to generate content (speak or move) at any time and send it to other users. In live interactive streaming, receiver cannot delay its playback as it loses the interaction with other participants. This results in a *tight* timing constraint for such application.

2.1.3. Comparison

P2P streaming applications depending on the type of the streaming content differ in some fundamental aspects and have various sets of requirements regarding delay, bandwidth and scaling. The distinctions between these mechanisms lead to different goals and challenges in the design of such applications. The main differences between them can be pointed out as follows:

- *Content availability:* In on-demand streaming applications, the full content is available at source, while in live streaming (whether interactive or non-interactive), the availability of the future content is limited. This limited availability of content in live streaming applications makes the design of such applications more challenging.

- *Timing constraint:* P2P streaming applications are sensitive to end-to-end delay; packets that incur a sender-to-receiver delay above a certain amount are essentially useless. The level of timing constraint which determines the amount of allowed buffering at the receiver, is different among types of streaming applications. Live P2P interactive streaming applications have a tight timing constraint and thus, very demanding about end-to-end delay. On the other hand, for non-interactive streaming content, such as live video broadcasting, the delay tolerance is typically higher, due to lack of interactions among users. Between live and on-demand applications, the timing constraint is tighter in the former. In live streaming, the shorter the end-to-end delay is, the more lively the stream is perceived by receivers (referred to as liveliness in [16]). In on-demand streaming, liveliness is simply irrelevant because the stream is already pre-recorded.

- *Playing time:* In P2P on-demand streaming applications, users can watch the video or listen to the audio at any arbitrary time. While, in live P2P streaming applications (either non-interactive or interactive), all participants have a loosely synchronized viewing/listening time.

- *Scalability:* Ability to scale to a large group is often more of an issue for live P2P non-interactive streaming applications compared to the other two streaming applications. Huge live events have lots of interested users watching them live and simultaneously. Whereas, for on-demand streaming this is not common to happen. Live interactive applications that involves multiple users such as video conferencing are often used by small to medium sized groups due to the nature of conferencing.

Table 1. presents characteristics of various streaming applications along with an instance of each class of applications.

2.2. Design of Live P2P Streaming: Approaches

The goal of Live P2P streaming applications is to deliver content to a large group of participating peers. The main components of such applications are overlay construction and content delivery. Peers form an overlay with some specific properties (*i.e.*, shape and connectivity) and distribute the content through the connections in the overlay. In such applications, each peer receives the content from one or multiple

Streaming Content Type	**Application**	**Example**	**Time-constraint**	**Limited content availability**	**Scalability**
On-demand	Video-On-Demand	Skycrapper	Loose	X	Medium
Live non-interactive	Live sport broadcasting	PPLive	Relatively loose	✓	High
Live interactive	Video conferencing	VSee [17]	Tight	✓	Low
	IP telephony	Skype	Tight	✓	Low

TABLE 1.: Comparing characteristics of streaming applications.

other peers referred as *parents*, and might forward the content to other peers called *children*.

The main issue in live P2P streaming application is how to deliver content to all interested peers. For this an overlay is constructed among peers which can be a single *tree*, *multiple trees* or a *mesh*. Further, content are delivered to all peers on top of the overlay, which can be performed in a *push* or *pull* fashion.

Several research studies proposed solutions for P2P live streaming that suggest different approaches for overlay construction and content delivery. Based on the characteristics of the overlay (*i.e.*, shape of the overlay) and the way the content is mapped to the overlay, existing solutions can be broadly classified into four approaches: *(i) single-tree-based*, *(i) multiple-tree-based*, *(ii) mesh-based*, and *(iv) hybrid*.

A simple way of delivery of content over a P2P network is to organize peers into a single source-rooted tree and push the content on top of that; typical examples include ESM [18] and Yoid [19]. In this approach, each peer has only one parent and receives the whole content from that parent. The single-tree-based approach is vulnerable to churn, a leave or failure of a peer, particularly departure of a parent which is close to the source may cause disruption in delivery of content to all the descendants of that peer. The tree thus has to be reconstructed frequently, which adds extra costs and overhead. Moreover, this approach has some generally known drawbacks such as limited scalability due to inability to utilize all peers' bandwidth and inability to accommodate heterogeneity of peers' bandwidth.

Multiple-tree-based approach tries to overcome the limitations of the single-tree-based approach. In such an approach, multiple trees is built between peers, each of the trees delivers a particular part of the streaming content. CoopNet [20] and Chunkyspread [21] are protocols based on the multiple-tree-based approach. In multiple-tree-based approach, the streaming content is divided into sub-streams or descriptions and then each sub-stream is mapped to a particular tree. While this approach can overcome some limitations of single-tree-based approach, the multiple-tree overlay is more complex to be constructed and maintained.

Mesh-based approach constructs a mesh over all participating peers and incorporates swarming on top of that for delivery of the content [4]. In this approach, each peer has multiple parents from whom it pushes the content and multiple children that serve the content to. The content is divided into equal sized blocks. Each peer explicitly informs its children of the block availability information and periodically requests blocks from its parents.

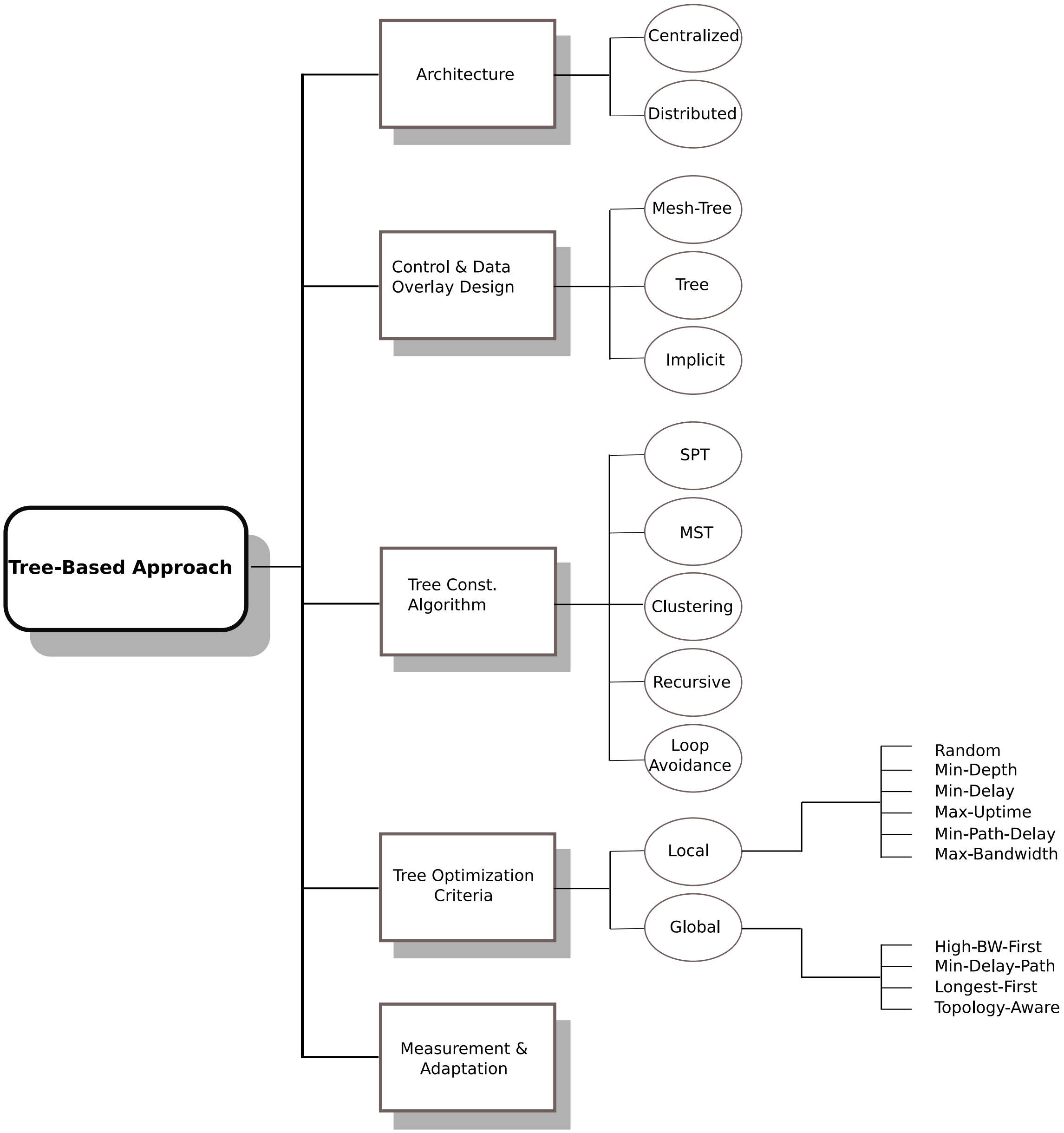
An alternative approach is hybrid which sits in the middle of the multiple-tree and swarm-based approaches. A hybrid approach is basically a mesh-tree or pull-push approach in which peers receive part of the content from their parent in a tree structure and pull the rest of the content from their mesh parent(s). Bullet [22] and mTreebone [23] are two protocols that incorporate such an approach.

2.2.1. Single-tree-based

The most popular and easiest approach for P2P streaming is single-tree-based approach [24]. This approach is commonly referred to as *application-level multicast*. The goal of single-tree-based approach is to construct a source-rooted tree across all peers. Streaming content is pushed into the single tree to reach to all peers; that is, when a peer with at least one child, receive a block of content it forwards the block to all of its children. Therefore, in the single-tree-based approach, all blocks of the stream follow the same tree to reach to all peers. This motivates construction and maintenance of an optimized tree-shape overlay.

In the single-tree-based approach, a newly joined peer initially contacts a central server or bootstrap node to find at least one existing peer. This existing peer can be either a parent for the new peer or an initial contact point for the new peer to run the protocol's tree construction algorithm to find its parent. The way the parent is selected (*i.e.*, the tree is constructed) differs from one protocol to another. Further, to maintain the content delivery tree, such a protocol must be able to detect any failure and partitioning, as peers join and leave the session [25]. In particular, if a peer fails or leaves, the overlay should be repaired quickly, otherwise the delivery of the content gets disrupted to all lower level peers in the sub-tree rooted at that peer. Protocols adopted different methods for tree maintenance, some relies on a central server to maintain the tree, while others form a control topology and let peers communicate over that [26]. Moreover, as network condition may change over time, some protocols provide adaptation mechanisms to improve the quality of the content delivery tree.

In general, the basic components of a single-tree-based P2P protocol as shown in Figure 1., consist of: *(i)* a centralized or distributed architecture for overlay construction and maintenance [26, 27], *(ii)* a tree-first, mesh-tree, or implicit design for control and data overlay, *(iii)* an algorithm for tree construction, *(iv)* performance criteria for tree optimization, and *(v)* an adaptation mechanism for coping with network dynamics. To provide a systematic insight into the design of different single-tree-based P2P protocols, we broadly categorize them according to the above components.



(a)

FIGURE 1.: An overview of the tree-based approach components along with the design choices of each component.

2.2.1.1. Architecture

Single-tree-based P2P protocols can differ in the architectural approaches adopted for the overlay construction and maintenance. Joining and leaving the session, peer failures, and variations in network conditions result in dynamics in the content delivery tree. Therefore, the approach adopted for the overlay construction and maintenance is a key issue in single-tree-based protocols. This determines the centralized or distributed architecture for construction and maintenance of the overlay tree. A centralized approach refers to performing all the functionalities regarding to tree construction, optimization and maintenance at a central server. The central server collects measurements from all peers. It allows the participating peers to build a source-rooted tree. The main advantages of a centralized architecture are its simplicity, being fast in computing the tree, its reliability in preventing tree partitions and loops. In events of peer departure of failure, the centralized architecture is also efficient in repairing the tree. However, this architecture lacks the scalability and has reliability issues as it is more vulnerable to a single point of failure.

In a distributed architecture, the load of construction and maintenance of the tree is distributed across all participating peers. Participating peers are responsible for finding their own or another peer's optimal position in the tree. Therefore, it is relatively more robust to failures as there is no single entity in the session, that its failure affects the entire group. Moreover, the distributed architecture is more scalable due to relaxing the need for a central server as a potential bottleneck. Although intuitively one might think that the distributed architecture for tree construction and maintenance is more suitable for scalable delivery of streaming content, there are still positive arguments for a centralized architecture [28, 29]. A fully distributed architecture causes excessive overheads and is not as optimal, efficient and quick in building an optimal tree. Further, a synchronized communication among the participating peers and thus, fast decision-making process are hard to achieve with distributed architecture [29]. There is a trade-off between simplicity and practicality of a centralized architecture versus robustness and scalability of a distributed architecture.

2.2.1.2. Control & Content Overlay Design

In distributed architectures, to facilitate the tree construction and maintenance, participating peers can be organized into two overlays: control and content overlay. Peers on the control overlay may probe each other to measure some performance metrics and periodically exchange update messages to identify and recover from sudden ungraceful departures. The content delivery overlay is usually a subset of the control overlay and identifies the paths for delivery of content from source to all peers. The content delivery overlay is a tree, while the control overlay might have a separate overlay mesh structure where peers have higher connectivity or it may share the same structure as the content delivery overlay. There are three different designs for control and content delivery overlay topology as follows: tree-first, mesh-tree, and implicit.

Tree-first: In the tree-first scheme, participating peers are organized into a tree for content delivery by selecting their parents from known peers (in the distributed or hybrid architectures) or by connecting to a peer assigned by the central server (in the centralized architectures). Peers on the tree may establish and maintain additional control connections to other participating peers to enhance the robustness of the tree in dynamic environment. These additional connections along with the content delivery tree form a control overlay mesh. Examples of protocols using this scheme are Yoid [19] and HM [30]. The tree-first scheme is simple and scalable due to the low communication overhead. It also gives direct control over the content delivery tree. However, this scheme requires running additional algorithms for loop avoidance and detection along with the partition detection to ensure that the overlay is indeed a tree [19].

Mesh-tree: In the mesh-tree scheme, the first step is constructing a mesh for control traffic and then a tree for content delivery. Participating peers first organize themselves into a mesh as the control overlay. The mesh is usually optimized for some performance criteria and dynamically adjusted to accommodate the underlying network changes. For instance, if a peer arrival or departure occurs, or some mesh connections experience congestion, the mesh may be reconstructed to adapt to the changes. The second step is the construction of a source-rooted tree for content delivery on top of the mesh. Participating peers run a well-known routing algorithms (*e.g.*, DVMRP [31]) on the control overlay to compute the content delivery tree in a distributed way. In the mesh-tree scheme, the quality of the tree depends on the quality of the mesh. Therefore, the challenge is to construct an efficient mesh, that is, each link in the mesh, has a good property defined by the protocol's specific optimization criteria (*e.g.*, . minimum delay or high bandwidth), while maintaining the mesh with low overhead (*i.e.*, each peer has a small number of neighbors in the mesh). One of the most popular protocols uses this scheme is Narada [18]. Protocols employ mesh-tree scheme are efficient for small peer population, but do not scale well beyond a few tens of peers [32]. The Mesh-tree is more complex than the tree-first scheme. However, it provides more resilience to peer departures and simplifies tree construction and maintenance as loop avoidance and detection are built-in mechanisms in well-known tree construction algorithms such as DVMRP.

Implicit: Implicit performing control and content delivery overlay at the same time is also possible and is called implicit scheme. In this approach a control overlay is constructed among the participating peers. The overlay for content delivery is implicitly determined by a set of routing rules for each content block. These rules leverage the specific properties of the control overlay to build the tree. In the implicit scheme, participating peers are usually grouped into clusters. Peers with a relatively close performance criteria (*e.g.*, topologically or latency-wise) are forming the same cluster. NICE [33] and Zigzag [16] are popular representatives employ this scheme. The advantage of this scheme is its flexibility and scalability as each peer need to maintain information about only a small subset of participating peers. However, the resulting tree might not be optimal.

2.2.1.3. Tree Construction Algorithm

A tree construction algorithm typically involves a solution to a graph theory problem. That is, given a certain graph $G = (V, E)$ (*i.e.*, V and E represents all peers and the overlay connections, respectively), and certain constraint on each peer (*i.e.*, peer degree), the problem is constructing a source-rooted tree spanning all peers. The tree is constructed with a goal of optimizing a performance criteria; *e.g.*, minimizing the delay/hop-count from source to each peer, minimizing the total number of hops, or the cumulative end-to-end delay of delivery of content to all peers. Existing algorithms for tree construction are as follows:

Shortest Path: The aim of this algorithm is to construct degree constraint minimum diameter spanning tree. A Shortest Path Tree (SPT) algorithm such as Dijkstra constructs a tree in which the path from source to each peer has the minimum cost, where cost is the protocol specific optimization criteria such as hop count or delay. An SPT or one of its variants is commonly used by various proposed tree-based protocols such as TAG [34].

Minimum Spanning Tree: This algorithm tries to build a low cost tree without considering degree constraint of peers. In a graph with a cost associated with each edge, a minimum spanning tree (MST) is a tree with minimum sum of total cost connecting all peers. MST is commonly used by centralized approaches such as ALMI [28] and HBM [35] to construct a low cost tree that is not rooted at any particular peer and all peers use the same tree to distribute their content. Cost can be the round-trip-time delay between two connected peers or hop counts between source to peers. MST usually contains peers with high degree which will be overloaded. Finding a degree-constraint minimum spanning tree is NP-complete [36]. Therefore, in practice many single-tree-based protocols do not seek to find the minimum cost spanning tree, rather try to construct a near-optimal spanning tree that has bounded peer degree.

Clustering: Clustering algorithm divides peers into different clusters based on some desired criteria (*e.g.*, topologically close). In each cluster, a peer as a head is responsible for receiving the content from outside of its cluster and sending that to all the members of the corresponding cluster. Some protocols such as ZigZag [16] and NICE [33] deployed such a clustering algorithm for tree construction. Clustering algorithm has low computation overhead, however, the resulting tree might not be optimal and some peers might have very high degree.

Recursive: Recursive algorithm mostly used in distributed protocols, where each peer upon joining or rejoining recursively traverses the existing tree from source to find its appropriate parent and position in the tree. To find a suitable parent, peers can traverse the tree and repeat the top-down process in various ways such as breadth-first-search, depth-first-search or by some redirection hints from another peer (*e.g.*, A peer first contacts the source, chooses the best peer among the source children, and repeats this top-down process until it finds an appropriate parent). The position of each peer is decided based on desired criteria of the protocol. For instance, when the criteria of tree construction is building a minimum depth tree, each peer traverses the tree from source in breadth-first-search fashion until it finds an existing peer with empty slot to accept it as a new child. SpreadIt [37] and Overcast [38] use recursive algorithm to build their desired tree.

Loop avoidance: Some existing solutions adapt loop avoidance algorithm to construct a tree, in which each peer with some methods detects whether choosing a specific parent creates a loop or not. One method for detecting loops is through the *root path* [19]. The root path is the set of peers in the path from source to each peer which is analogous to AS path in BGP. If a peer finds that the root path from its potential parent contains itself, it should choose another parent otherwise a loop will be formed in the overlay. This method in incorporated in Yoid [19].

2.2.1.4. Tree Optimization Criteria

In the single-tree-based P2P applications, depending on the design goal(s) of the proposed protocol, constructed tree can be optimized for various local or global criteria. Such criteria directly affect the resulting shape of the tree. For instance, a protocol that aims in constructing a tree in which high bandwidth peers are located at the top levels, results in a tree with low depth. Local criteria are the ones that are optimized for each peer independently, whereas, global optimization criteria of tree affects the overall shape of the tree and its characteristics. Global optimization might require some tree reconstructions and preemptions. Single-tree-based protocols might be optimized for one or multiple of these criteria. Next, we explain these local and global optimization criteria.

- **Local optimization criteria**
  - *Non/random:* There is no specific optimization criteria, and tree is constructed at random. This criteria leads to the most efficient with the lowest overhead tree construction. However, the resulting tree might have a large depth [39].
  - *Max-Bandwidth:* Each peer tries to find a parent with maximum available bandwidth to ensure good quality and fast delivery of the content. Overcast [38] employs this optimization criteria with the goal of maximizing the bandwidth of the path from the source to all peers.
  - *Min-Delay:* Each peer finds a closest parent in terms of end-to-end delay. The intuition behind this criteria is to potentially minimize the delay from source in a light-weight fashion.

  - *Min-Path-Delay:* Peers try to find a potential parent with minimum delay from source through the overlay tree to achieve smaller buffer size and better liveliness. Clearly, this criteria has large overhead.
  - *Min-Depth:* Peers find a parent with minimum depth in the tree. This criteria potentially reduces the likelihood of bottleneck among upstream connections and achieve better robustness. The reason is that a smaller number of ancestors reduces the probability of late arrival of the packets due to the congestion occurrence in any upstream connections and it also decreases the probability of disruption in delivery of content the ancestors.
  - *Max-Uptime:* To decrease the likelihood of parent departure and disruption in delivery of the content, peers try to find a parent with maximum uptime or session time.

- **Global optimization criteria**
  - *High-Bandwidth-First:* Peers with higher outgoing access link bandwidth are placed in the top levels of the tree [40]. In such a tree, peers are organized from high to low levels in a non-increasing order of their outgoing access link bandwidth; that is, peers do not have more bandwidth than any peer higher up in the tree. This criteria allows later arriving peers to preempt the positions of existing peers with smaller bandwidth. It can achieve a minimum depth tree at any time but needs frequent preemptions

and reconnections between peers to maintain such a globally ordered hierarchies. The departure of a peer may impose very high overhead on the descendants to reconstruct the tree with such a criteria.

- *Min-Delay-Path:* This criteria tends to minimize the delay of each path from source to any peer. The shortest path tree construction algorithm can globally achieve that by computing a shortest path tree over some existing edges.
- *Longest-First:* This criteria builds resilient trees by minimizing the number of interruptions in delivery of content resulted from departures of peers. It puts the longest-lived peers in higher level of the tree; the intuition behind this is that when the peers' lifetime follows a heavy-tailed distribution, the older peers generally tend to stay longer than newer peers [39].
- *Topology-aware:* Tree is constructed and maintained in a way that peers that are closer to each other in terms of underlying network topology connect to each other. This criteria can potentially reduce the delay between peers and decrease duplicate network resource usage [34].

2.2.1.5. Overlay-based Measurement and Adaptation

A tree-based P2P application must continually adapt itself to the variation of network environment and to a new set of participating peers due to the joining of new peers or departure of existing peers. Internet is a highly dynamic environment and prone to unpredictable congestion, partitions and flash crowds. Thus, an efficient tree at some point may become very inefficient after a period of time. Moreover, the arrival or departure of peers might result in a completely different efficient and optimal tree. Therefore, protocols should be able to discover network variations by means of frequent measurements, adjust to a new set of participating peers and adaptively alter the tree to reflect any changes.

Many measurement techniques to capture overlay connections' characteristics have been proposed [41, 42, 43, 44, 45]; they usually send messages to estimate the desired criteria such as end-to-end delay or available bandwidth between participating peers. In these techniques, the delay can be derived by RTT (round-trip time) and a 10-KB TCP probing are used for available bandwidth estimation. Unfortunately, available bandwidth is highly fluctuating as it depends on access link bandwidth and cross traffic which is highly variable in the Internet. Therefore, frequent verifying the validity of a measured available bandwidth causes high overhead. Some techniques [46, 47] try to estimate the bottleneck bandwidth which is stable and accurate. Bottleneck bandwidth is the upper bound for available bandwidth. While bandwidth measurement is an active area of research [48, 49], accurate results generally require the transfer of a large amount of data to gain confidence in the results and due to its high variability over short time periods it's frequent measurements leads to a high overhead.

After detecting any changes, tree-based protocols adapt the tree overlay to reflect those variations and improve optimality of the tree. Many tree-based protocols incorporate adaptation to the tree in a local way by allowing peers to dynamically change to a better parent instead of globally change the tree structure. For example, Overcast [38] estimates the available bandwidth between a peer and its potential parent. In addition, each peer periodically check its position in the tree by measuring the bandwidth to its current neighbors and ancestors. Based on the new measurements peers may switch parent. Note that, excessive adaptations make the tree overlay unstable.

Table 2. compares a number of tree-based protocols with respect to various characteristics described in this Subsection 2.2.1..

2.2.1.6. Case Studies

In this section, we cover some of the protocols mentioned in Table 2., to get a better understanding of how they perform. We look at Narada [18], ALMI [28], OverCast [38] and NICE [33] and ZigZag [16].

Narada [18]

Narada is one of the earliest tree-based P2P protocols for many-to-many streaming applications. It incorporates a distributed mesh-first scheme for tree construction, maintenance. Peers initially form a well-connected overlay mesh as a control infrastructure. Further, they run the DVMRP routing algorithm to construct a

Protocol	Architecture	Control & Data Overlay Design	Tree Construction Algorithm	Optimization Criteria	Adaptation	Goal
ALMI [28]	C	NA	MST	Global Min-Total-Delay	✓	Minimizes total delay to all peers
AMCAST [50]	C	NA	MST	Global Min-Total-Delay	X	Minimizes total delay to all peers
BTP [50]	D	Tree-first	Recursive	Local Min-Delay	✓	Minimizes delay of delivery path to each peer
HBM [35]	C	NA	MST	Global Min-Total-Delay	✓	Minimizes total delay to all peers
Narada [18]	D	Mesh-first	SPT	Global Min-Delay-Path	✓	Minimizes delay of delivery path to each peer
NICE [33]	D	Implicit	Clustering	Local Min-Delay	✓	- Maximizes scalability by lowering overhead on peers - Minimizes delay between peers
OMNI [51]	D	Tree-first	SPT	Global Min-Delay-Path	✓	Minimizes delay of delivery path to each peer
Overcast [38]	D	Tree-first	Recursive	Local Max-BW	✓	Maximizes bandwidth from source to peers
ProBaSS [52]	D	Tree-first	Recursive	Local Min-Delay	X	Minimizes delay between peers
RITA [53]	D	Tree-first	Loop Avoidance	Local Min-Delay	✓	- Minimizes delay between peers - Efficient & scalable adaptation mechanism
Scattercast [54]	D	Mesh-first	SPT	Global Min-Delay-Path	✓	Globally minimizes delay from source to peers
SpreadIt [37]	D	Tree-first	Recursive	Local Min-Delay	X	Minimizes delay between peers
TAG [34]	D	Tree-first	Recursive	Global Topology-aware	✓	Minimizes stress & stretch
TBCP [55]	D	Tree-first	Recursive	Local Min-Delay	X	Minimizes delay between peers
Yoid [19]	D	Tree-first	Loop Avoidance	Local Min-Delay	✓	- Minimizes delay between peers - Minimizes loss rate
ZigZag [16]	D	Implicit	Clustering	Local Min-Delay	✓	- Maximizes scalability by lowering the overhead on each peer - Minimizes delay between peers

TABLE 2.: Taxonomy of some of the tree-based P2P streaming protocols.

spanning source-rooted tree over the mesh for delivery of the content. Peers through a gossip-based peer discovery protocol keep an updated list of all the existing peers to detect overlay partitions and to select the best paths for delivery of the content. When a new peer joins the session, it retrieves a list of existing peers from a bootstrap node and sends a neighbor-request to them. Peers establish one or more mesh connections at random. Once a peer connects to the mesh, it starts exchanging periodic update messages with its neighbors. These update messages are propagated through the mesh to all the other participating peers.

The goal of Narada is to construct a shortest path tree for each peer as a source (source-specific) in which the delay from source to each peer is minimum. This tree is a spanning tree of the mesh. Therefore, the quality of the tree depends on the quality of mesh connections. Due to random connections between peers the mesh initially is not efficient. To construct an efficient mesh and also adapt to the variation of network conditions, peers periodically optimize their connections. Towards that, peers continuously probe other random participating peers, and may add or drop connections depending on the perceived gain in *latency*. This periodic probing also helps them detect a participating peer failure which further will be propagated throughout the mesh.

Overcast [38]

Overcast tries to construct a tree in which the bandwidth from source to each peer is maximized. Overcast places peers as far away from source as possible without sacrificing bandwidth. The tree is constructed with the recursive algorithm in a distributed fashion. Peers upon joining contact the source and choose the source as their parent. A series of rounds will begin in which they decide where on the tree their position is. In each round, peers measure their available bandwidth to their parent and to their parent's children by considering the time to download a 10KB data. If the bandwidth through any of the children is about as high as the bandwidth to their current parent, then that child becomes their new parent and a new round commences. In case, there are multiple suitable potential parents (bandwidth differences within 10%), the child with minimum number of hops reported by traceroute will be chosen as a new parent.

To adapt to network variation and optimize the tree, peers periodically reevaluate their position in the tree by measuring the available bandwidth between themselves and their parent, grandparent and all siblings. Based on the new measurements, peers may switch parent. To maintain the tree in a distributed way, peers keep an ancestor list to reconnect to the tree in case of a parent departure. When a peer detects that its parent is unreachable, it will connect to its grandparent or continue to move up its ancestry until finds a reachable peer.

ALMI [28]/HBM [35]

ALMI and HBM aim to build a Min-Total-Delay tree in a centralized tree-first fashion. In both ALMI and HBM, participating peers are organized into a tree, which is formed as a minimum spanning tree (MST) using delay as the metric. The only difference between ALMI and HBM is that ALMI builds the tree using only partial knowledge of inter-peer delay measurements, whereas, HBM requires the measured delay information between all peers (full knowledge).

A new peer will join the tree by contacting the bootstrap node. The bootstrap node then refers the new peer to a randomly-selected participating peer. The bootstrap node ensures the efficiency of the tree by periodically calculating a minimum spanning tree based on the measurement updates received from all the participating peers. To collect measurements the bootstrap essentially instructs each peer to periodically send probes to a constant number of other peers and measure the round trip delay. This delay measurements serve as the costs used to periodically re-calculate a new minimum spanning tree.

NICE [33]

NICE focuses on implicitly building a Min-Delay tree in a distributed fashion while maintaining the overlay with low overhead. It organizes peers into multi-level hierarchy of clusters with bounded maximum and minimum size. While constructing the hierarchy, nearby peers (in terms of end-to-end delay) are mapped to the same cluster. Cluster-mates keep detailed state about each other. A peer that has the minimum maximum distance to all other peers in a cluster is the leader of that cluster. The source-rooted tree for content delivery is implicitly defined from the hierarchy. A leader of each cluster in each level is responsible for communicating with higher levels and delivery of the content to its cluster-mates. In each cluster, the control overlay is a clique, while the content delivery overlay is a star.

Upon joining, a new peer must locate the nearest cluster to itself. This is done in a recursive way, such that the new peer first contacts the highest level peer and then requests from it a list of peers in the next lower level. The new peer should probe each of the them and selects the closest one. Further, the closest peer will inform the new peer the list of its cluster-mates in the next-lower level. With this method, the new peer can iteratively find its level 0 cluster. To maintain the hierarchy and thus the tree, peers periodically send heartbeat messages to their cluster-mates. Leaders of each cluster should also keep state about their higher level cluster-mates. They are also responsible for maintaining proper cluster size and thus apply the splitting or the merging algorithm when needed.

To adapt to network variations and changes due to peers arrival and departure in NICE, each peer in any level, periodically probes all cluster leaders of its level to identify the closest peer to itself in other clusters. If there is a closer cluster, it leaves the current cluster and switches to the new cluster. Leader selection also periodically should be recalculated and if necessary, a new leader will be elected.

By clustering the participating peers, the control overhead is effectively reduced, since a peer must maintain state for a limited number of its cluster-mates. This lets NICE scale better than a mesh-tree protocols (*i.e.*, Narada). However, in NICE, the some peers requires to have an unbounded bandwidth (*e.g.*, leaders of higher levels) and a very high overhead. Moreover, the large disruption time in case of cluster leader failure is another issue in NICE.

ZigZag [16]

ZigZag is very similar to NICE. It focuses on a large scale P2P streaming application with the goals of low delay from source to all peers, quick recovery from failures, and small overlay maintenance overhead. It tries to address some of the issues of NICE. ZigZag does so by splitting the role of the leader over two different entities, the head and the foreign-head. When a peer wants to join, it submits a request to the source, then this peer traverses along the tree downward from the source until finds the closest cluster to join by probing each. On each cluster, the foreign head is responsible for getting the content and transmitting it to all its non-head cluster-mates, whereas, the head is responsible for overlay maintenance. Thus, unlike NICE, the control overlay in ZIGZAG does not infer a content delivery tree.

Zigzag has been shown to have a better performance than NICE due to its method of constructing tree: *(i)* the worst case degree of NICE is $O(logN)$, while it is bounded by a constant in Zigzag and *(ii)* failure recovery in Zigzag is more efficient that that of NICE, *i.e.*, NICE requires $O(logN)$ peers to reconnect to the overlay while overhead in ZIGZAG is upper bounded by a constant. Both NICE and Zigzag require that most of the peers have a guaranteed minimum outbound bandwidth, ı.e. an integer multiple of the stream bandwidth. Moreover, both of these complex protocols have to deal with cluster splits and merges, leader selections and peer departures. They are not optimized for a high rate of churn, joining the session requires $O(log(N))$ messages, where N is the number of participating peers, and disruptions in the tree due to peer failures can take up to 30 seconds to deal with. To maintain the structure of trees they require lots of control traffic (*i.e.*, periodic heartbeat messages between peers in each cluster). Moreover, the number of peers that partition from the tree and need to reconnect due to a failure is significantly large (*i.e.*, $O(logN)$ in Nice and a $O(k^2)$ in ZIGZAG).

2.2.2. Multiple-tree-based

More recently, multiple-tree-based approach [29, 20, 56] for P2P streaming have been proposed to increase the overall resiliency and efficiency of the single-tree-based approach. In the single-tree-based approach, the non-leaf peers deliver the content, while the leaf peers only receive the content. Therefore, the upload bandwidths of the leaf peers are not utilized for content delivery. Multiple-tree protocols such as CoopNet [29] and Splitstream [56] overcome this inefficiency by constructing multiple trees and distributing part of the content in each tree. Any peer could be an interior in one or multiple of the multicast trees, and contribute in delivery of the streaming content.

The stream is split into smaller sub-streams or descriptions (if encoded with MDC or LC) with equal bit rate. In such an approach, multiple source-rooted trees with desired properties are formed and each sub-stream is simply pushed to a particular tree by source. Each participating peer receives sub-streams from specific parent and serves multiple children. Each peer decides a proper number of trees to join based on its incoming access link bandwidth. The out-degree of each peer determines the number of children that it can support. Participating peers might contribute (*i.e.*, be *fertile* and have children) in all the trees that they have joined or in a subset of them. The policy adapted for allocation of out-degree or distribution of the number of children of each peer across various trees can affect the performance of such an approach.

Similar to the single-tree-based approach, the multiple-tree-based approach can be fully centralized [29] or it can have a distributed architecture [21]. Besides the architectural choice, other components of multiple-tree-based approach include optimization criteria and algorithm for construction of each tree, maintaining each tree and adaptation mechanism to improve trees in case of changes in network conditions. These components and their design choices are basically similar to those of the single-tree-based approach discussed earlier in Subsection 2.2.1.. However, the new component in multiple-tree-based compared to single-tree-based approach, is the *out-degree allocation policy*. In the single-tree-based approach, depending on the out-degree of peers, they are either intermediates and have children or leaves without any children, whereas in the multiple-tree-based approach, peers can allocate their out-degree across various trees. In this section, we focus on this new challenge and component of the multiple-tree-based approach which is the out-degree allocation policy.

2.2.2.1. Out-degree Allocation Policy

In the multiple-tree-based approach, each participating peer can be fertile and have children in trees that it has joined. Based on each peer's out-degree (*outdeg*), the peer can be fertile in all of its joined tree or a sub-set of them. The new challenge in multiple-tree-based approach, is how to allocate *slots* within each peer's out-degree into various trees. Existing multiple-tree-based protocols propose various out-degree allocation policies as follows:

- *Random*: The motivation for random allocation policy is building diverse trees with low overhead. In random trees, while each peer may have upto *outdeg* children, the distribution of the number of children of a peer across different trees is random without any constraint. Therefore, within the constraint imposed by *outdeg*, peers can accept children in any trees. The random policy is incorporated in Chunkyspread [21].

- *Interior-disjoint*: Each peer can be fertile in only one tree and in the rest of the trees is a leaf. CoopNet [29] applied this policy to allocate peers' contribution across various trees. This results in the trees with minimum depth which have desirable properties such as minimizing the number of affected peers at each peer departure and minimizing the propagated effect of bandwidth fluctuation from upstream connections to lower levels as described in Subsection 2.2.1.. Moreover, interior-disjoint policy increases the stability of trees and simplifies tree maintenance in presence of churn, as the departure of a peer only partitions one tree.

- *Balance*: To have balance trees in which the number of peers that each tree can accommodate is equal, resources should be equally divided among trees; that is the sum of allocated slots by peers in each tree should be roughly equal. Therefore, the choice of which tree a certain peer should be fertile and how many slots it should allocate at depends on the number of slots at each tree. CoopNet tries to build balance trees.

- *One-child-per-Tree*: Peers have one child in each tree, which is analogous to minimum breadth trees [57]. These trees are stable and easy to manage but result in very long trees that suffer from long delay for content delivery and maximizing the probability of bottleneck propagation from upstream levels to lower levels.

- *Local-Balanced*: Every peer allocates equal number of slots into its fertile trees [57].

Protocols can apply more than one of the above policies to allocate peers' slots across various trees. For instance, CoopNet [29] tries to build balance and interior-disjoint trees by letting a peer to be fertile in only one tree which has the minimum number of fertile peers. They further, compare the performance of interior-disjoint balance allocation policy with random policy and showed that the interior-disjoint balance policy performs significantly better because it is able to construct shorter and also more diverse trees. However, building interior-disjoint trees is complex specially in heterogeneous environment in which peers have various outgoing access link bandwidth or correspondingly out-degree. Moreover, authors in [57], through analysis and simulations, discussed that interior-disjoint policy imposes some limitations on tree reconstruction in presence of churn. In fact, they showed that limiting the number of fertile trees for each peer although results in shorter trees, it increases the reconnection attempt failures for disconnected peers through ancestor departure. Random allocation policy minimizes the probability of experiencing reconnection attempt failure for a disconnected peer.

Table 3. compares a number of multiple-tree-based protocols with respect to various characteristics described in this Subsection.

2.2.2.2. Case Studies

CoopNet [29]

CoopNet focuses on organizing participating peers into multiple diverse trees to minimize the effect of churn and effectively utilize available resources in the streaming session. The goal of tree construction is to maintain multiple stable minimum depth trees with interior-disjoint and balance policy. CoopNet relies on existence of a central server for overlay construction and management. Streaming content is encoded with MDC and divided into descriptions. Each description is simply pushed through a separate tree. A newly joining peer first contacts the central server and informs its available out-going and in-coming bandwidth to determine its out-degree and the number of trees that it wants to join, respectively. The server first decides the tree in which it is going to be fertile; in all other trees the peer is a leaf node to achieve stability in presence of churn. The new peer is added as fertile peer to the tree that has the minimum number of fertile peers to keep the population of fertile peers balanced among different trees. To maintain minimum depth trees, the new fertile peer is placed as a child for the peer with the lowest depth that can accommodate a new child or has a child that is a leaf. In the latter case, the new peer replaces the leaf peer and the preempted leaf should rejoin the tree similar to a new leaf. When a fertile peer of a tree departs/fails, each one of its children as well as the sub-tree rooted at it are partitioned from the original tree, and should report the departed peer and contact the central server to rejoin the tree. Peers in such a partitioned sub-tree, initially wait for the root of the sub-tree to rejoin the tree. If the root is unable to join after a certain period of time, individual peers in a partitioned sub-tree independently contacts the server to rejoin the tree.CoopNet, employs an adaption mechanism based on MDC, in which the central server periodically gathers participating peers reception information, and feed this information into the MDC optimizer to adapt to the network dynamics and group size. Thus, MDC continuously adapts to the incidence of packet loss in the network, with more redundancy added when packet loss is frequent and vice versa.

In CoopNet, a tree can always accept a new internal node. However, in presence of churn, a tree could become *saturated* and thus unable to accept any new leaf node. This occurs when a tree loses a fraction of its internal nodes within a short period of time which reduces the number of leaf nodes that it can accommodate. In this case, the number of internal nodes at different trees becomes imbalanced, where spare slots for leaf nodes are available on other trees but they can not be used to resolve the problem of the saturated tree.

Splitstream [56]

Splitstream is very similar to CoopNet and advocates the use of multiple-trees for live streaming applications. However, unlike CoopNet, Splitstream constructs the overlay in a distributed fashion, on top of the Scribe protocol [58] which is in turn based on Pastry [59], a DHT P2P substrate. Using Scribe relaxes the need for a resourceful central server. However, constructing interior-disjoint trees in a distributed fashion adds extra complexity and control overhead. For instance, in case a peer that has reached its out-degree limit receives a join request from a child, it should disconnect one of its existing children and accept the new one. The disconnected child then seeks to locate a new parent in multiple extra steps which might not be successful without violating the interior-disjoint policy. Therefore, in Splitstream, a peer might be assigned more children than it can handle, trying to avoid that either requires sacrificing interior-peer-disjointness or leads to a long disruption in receiving the streaming content for some peers.

Chunkyspread [21]

Chunkyspread is another multiple-tree-based protocol which tries to build multiple trees with random allocation policy in a fully distributed fashion. The stream is divided into various sub-streams with equal bitrate. Each sub-stream is pushed over a separate tree, while trees are random and not necessarily node-disjoint. To facilitate the construction of multiple-trees, Chunkyspread forms a well-connected random mesh by a continuously running distributed random-walk algorithm. Chunkyspread gives peers with higher outgoing access link bandwidth proportionally higher peer degree to potentially have more children.

Trees are constructed through a loop avoidance mechanism as described in Subsection 2.2.1.. Essentially, each peer tries to find a parent for each sub-stream without forming a loop. Chunkyspread avoids and detects loops by using bloom-filter in the data packets. Peers advertise the bloom filters they receive for every sub-stream to their neighbors. A given peer does not select a neighbor as a sub-stream parent if the peer itself appears in the neighbor's received bloom filter. Loop avoidance and detection add extra overhead and complexity for Chunkyspread.

Each peer has an accepted range for the number of children that it can serve and a target out-degree. Individual peers by considering the target out-degree, the current number of children and per-sub-stream bloom filters advertised by their neighbors, determine which neighbor would make appropriate parent for each sub-stream. To prevent from overloading a peer, each peer periodically checks to see if it has an overloaded parent, and an underloaded neighbor, and if so attempts to switch parents. Chunkyspread does not perform any adaptation to changes in network conditions such as fluctuation of available bandwidth or increase in loss rate and as long as parents' load is in their accepted range, children switch parents only to improve relative latency between sub-streams.

Upon failure or departure of a participating peer, only the immediate connected children try to find an appropriate parent for the corresponding sub-stream. During this recovery period, all the descendants of the departed peer are also disconnected from the corresponding tree.

CoolStreaming [60]

The new version of Coolstreaming, employs a multiple-tree-based approach for streaming of live content. Similar to Chunkyspread, the multiple-trees formed in Coolstreaming, are random and the streaming content is divided into sub-streams with equal bitrate. Membership management is through a gossiping mechanism. Peers initially form a random mesh by discovering each other through gossip messages. Each peer tries to find a parent for each sub-streams from the set of its neighbors in the mesh. Once a peer select a parent for a particular sub-stream, parents continue pushing the sub-stream to the peer.

Similar to Chunkyspread, trees are constructed through a loop avoidance mechanism but in a different way. Coolstreaming avoids loops by checking the latest timestamp available at neighbors for each sub-stream. If the largest timestamp available at a neighbor for a specific sub-stream is larger than the one available at the peer, the neighbor is closer to the source in the path of the corresponding sub-stream and can be a potential parent for that sub-stream. Among the potential parents for a particular sub-stream, each peer tries to select the one that is not considerably lagging behind other neighbors in terms of the largest available timestamp for any sub-stream.

In Coolstreaming, a parent will not voluntarily drop a child, therefore, it is upto the children to dynamically monitor the incoming bandwidth of their parent and trigger any parent switching if necessary. Parent switching is performed when

Protocol	Architecture	Tree Construction Algorithm	Optimization Criteria	Allocation Policy	Adaptation	Degree Constraint	Goal
Chunkyspread [21]	D	LA	Local Random & Min-Path-Delay	Random	✓	✓	- Improving relative delay of sub-streams - Overlay construction & maintenance
CoolStreaming [60]	D	LA	Local Random	Random	✓	✓	- Overlay construction & maintenance
CoopNet [29]	C	R	Local Min-Depth	Interior-disjoint & Balance	✓	X	Stable short trees
DAGSter [61]	C	R	Global Min-Depth	Random	✓	X	Stable trees
Splitstream [56]	D	R	Local Min-Delay	Interior-disjoint & Balance	X	✓	Stable short trees

TABLE 3.: Taxonomy of some of the multiple-tree-based P2P streaming protocols. LA: Loop Avoidance, R: Recursive

a sub-stream pushed by a parent is lagging behind other sub-streams at the child peer or other sub-stream among neighbors. Such peer adaptation and switching can potentially cause stream disruption and the instability of the overlay topology.

2.2.3. Mesh-based Approach

An alternative approach for P2P one-to-many content delivery is mesh-based. In this approach, participating peers form a connected mesh over which they incorporate swarming (*i.e.*, pull-based) content delivery. In this approach, each peer receives content from multiple parent and serves content to multiple children. In contrast to the tree-based, the mesh-based approach does not need to construct and maintain an explicit overlay structure for delivery of the content to all peers. Rather than constantly repairing a tree for delivery of the content in a hugely dynamic P2P environment, mesh-based approach uses the availability of content to guide the content flow. In such an approach, there is no pre-defined mapping of the content to connections, the content mapping to each parent is dynamic and locally decided at each peer.

Swarming content delivery enables participating peers to contribute their resources (*i.e.*, outgoing bandwidth) more effectively which in turn improves the utilization of available resources among peers, and leads to a better scaling property for the mesh-based approach. Swarming content delivery couples push content reporting with pull content requesting. Based on the availability information of the content at each parent, each peer as a child requests different blocks from parents. The blocks requested by each peer from a parent are determined by a *block scheduling* algorithm based on the available content and bandwidth from the parent. Block scheduling is the key component of the mesh-based approach, it aims to utilize available bandwidth from individual parents in order to maximize the received quality in the streaming context and minimize the total delivery time in the file distribution context.

There exists a few recent studies that have proposed a mesh-based P2P protocol which incorporate a variety of scheduling schemes, construct random or biased mesh and consider naive or intelligent source. Each of these protocols tries to achieve certain goals and maintain specific properties. To classify existing/possible mesh-based protocols for live P2P we first bring an overview of mesh-based P2P approach and further,

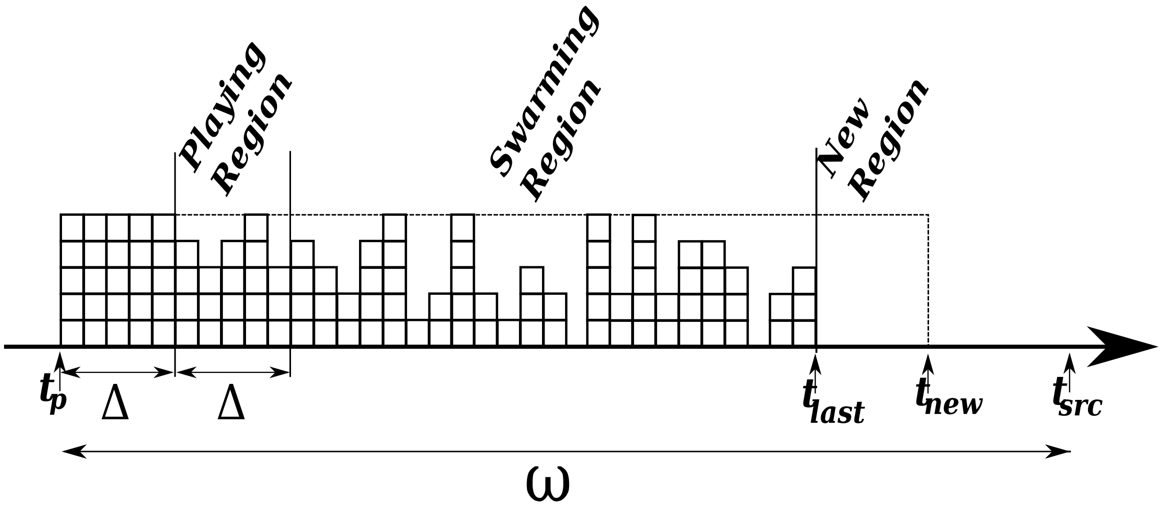


FIGURE 2.: Buffer state at a scheduling event in a peer.

discuss the main components of the mesh-based approach which are:*(i)* overlay mesh construction and maintenance, and *(ii)* block scheduling scheme.

Overview- In the mesh-based approach, participating peers form a connected mesh over which they incorporate swarming (*i.e.*, pull-based) content delivery. In General, each peer discovers other participating peers, through a centralized bootstrapping or a distributed mechanism. Each peer has multiple parents that receives content from and multiple children that provides content to. Note that the pairwise connections in the mesh-based approach, can be used for content delivery in both bidirectional or unidirectional fashion. To effectively incorporate swarming into P2P streaming and absorb any out-of-order delivery of blocks, each participating peer requires to maintain ω seconds worth of buffering. This implies that peers' should delay their playout time ω seconds behind source playout time (Figure 2.). This delay continuously provides ω seconds worth of content that can be used by peers for swarming. Further, to ensure in-time delivery of blocks, each block should be delivered to all peers within ω seconds from its generation time.

Pull-based content delivery is the key component of the mesh-based approach. As a parent, each peer progressively reports its new or available blocks to all of its child peers. Based on this availability information at each parent, each peer as a child determines a sub-set of blocks that should be requested (*i.e.*, pulled) from each parent. The requested blocks from individual parents are determined by a *block scheduling* algorithm at each child. Each parent simply delivers requested blocks by individual children in the provided order and at the rate that is determined by the available bandwidth between parent and the corresponding child.

2.2.3.1. Mesh Construction and Maintenance

An overlay mesh with the desired properties of simplicity, robustness and efficiency can be constructed in various ways. Existing mesh construction methods, try to achieve all or some of the above properties. We now describe methods of mesh construction along with their negative and positive properties.

- *Random*: In a random mesh overlay, each peer randomly selects its neighbors from the pool of known peers. The random mesh overlay is simple to construct with low overhead. Moreover, it is highly robust to partitions. However, random mesh overlays generally lack locality awareness, trying to improve the locality awareness in random mesh lead to network partition, which can not be prevented or even detected [62, 63].

- *Biased*: Each peer chooses the peer which has specific criteria as neighbor among the ones it is aware of. The criteria for neighbor selection can be the one with minimum delay or topologically close. While this overlay might be network efficient, it generally does not guarantee connectivity [62].

- *Best-Random-Parents*: A peer might choose some neighbors at random and the rest biased. This overlay provides a near-efficient mesh with probabilistic connectivity properties [62].

- *Loop-free or Directed Acyclic Graph (DAG)*: A DAG enforces a partial order among the peers and forbids loops. It has the following property: In a directed acyclic overlay with one source, if each peer (except the source and its direct children) has at least k neighbors, then the removal of any $k-1$ non-source peers does not cause any remaining peers to be disconnected from the source.

Another dimension in the mesh construction, is the relationship between connected peers. Essentially, the relationship between connected peers can be *bidirectional* or *unidirectional*, that is a mesh can be *directed* or *undirected.* In a directed mesh, connected peers have a parent-child relationship and the flow of the content is from a parent to a child. On the other hand, in the undirected mesh, the relationship between connected peers is bi-directional; there is no parent-child relationship between each two connected peers.

2.2.3.2. Block Scheduling

The key component of mesh-based content delivery is a *block scheduling* scheme at individual peers which determines the sub-set of blocks that should be pulled from each parent. Block scheduling can be performed periodically or in an event-driven fashion. In the periodic scheduling [64], the block scheduling function is invoked once per Δ seconds. In the event-driven scheduling [65], the invocation of the scheduling function is triggered by an event when a parent delivers all of the requested blocks and becomes idle. In the occurrence of such an event, the scheduling function invokes and determines blocks that should be requested from any idle parents. Block scheduling considers blocks within its current window of ω seconds (buffer) that should be pulled from parents in the current interval. The timestamp of the blocks in the current buffer falls within the following range [t_p+Δ, t_p+Δ+ω] where t_p is the peer's playout time. Figure 2. depicts a view of blocks with relevant timestamps (buffer state) for a peer at an scheduling event. t_p, t_{src}, t_{last} and t_{new} denote peer's and source's playout times, the largest available timestamp, and the largest reported timestamp in this scheduling event.

Block scheduling has two steps as follows: *(i) Block selection*, that determines the required blocks that are requested from parents. *(ii) Block assignment*, that allocates selected blocks to specific parents who can provide them.

Block Selection: Block selection should determine the missing but required blocks that still have sufficient time to be pulled from parents while considering the available blocks among parents. The block selection algorithm takes into account the aggregate incoming available bandwidth of the peer to determine the total number blocks that can be pulled during each interval or the block budget. Existing block selections are as follows:

- *Rare* or *Rand*: This algorithm selects all the blocks from the entire window using a rarest-first [64, 66] or random [65] strategy, respectively.
- *PRare* or *PRand*: This algorithm explicitly addresses the timing requirement by first requesting all the missing blocks that are in danger of being delayed beyond the deadline (*i.e.*, the blocks in the range of $[t_p{+}\Delta,\ t_p{+}2*\Delta]$ in Figure2.), and then using the remaining block budget to select rare/random blocks from the rest of the window $[t_p{+}2*\Delta,\ t_{new}]$ [67][1].
- *NPRare* or *NPRand*: This is a hybrid scheme [4] that gives priority to blocks just entered the window, then the ones that are in danger of being delayed, and

[1] [68] designs a similar selection scheme in which it probabilistically selects a missing block close to deadline.

finally uses any remaining budget to request a rare/random subset of blocks from the the rest of the window.

The output of the block selection algorithm is an ordered list of required blocks that are available among parents and should be requested.

Block Assignment: In this step, selected blocks should be mapped to requests from individual parents considering the blocks available at each parent. During each Δ seconds, each parent can send a limited number of blocks to the child that can be estimated by the available connection bandwidth from the parent to the child as $\frac{bw(i)*\Delta}{Blk_{Size}}$. The goal of block assignment is to utilize the incoming bandwidth of each peer during each interval to maximize the delivered quality. Therefore, it should fully utilize the available bandwidth of each parent by assigning appropriate number of blocks to each parent.

The existing solutions on block assignment are as follows:

- *Random:* Among the possible parents, assign the selected block to a random parent[65].

- *Min-Ratio:* Assign the selected block to a parent whom a smaller fraction of its block budget has been assigned so far [69, 64].

Protocol	**Overlay**	**Heterogeneity Support**	**Block Selection**	**Block Assignment**
AnySee [70]	Directed Biased, Location-Aware	X	*Rare*	*Min-Ratio*
Bitos [68]	Undirected Random	X	*PRare*	*Random*
BitTorrent	Undirected Random	X	*Rare*	*Random*
Chainsaw [65]	Undirected Random	X	*Random*	*Random*
DAGStream [63]	DAG, Location-Aware	X	X	X
DoNet [64]	Directed Random	X	*Rare*	*Min-Ratio*
GridMedia [71]	Random	✓	*Rare*	*Random*
PRIME [4]	Directed Random	✓	*NPRand*	*Min-Ratio*
Pulse [66]	Undirected Random	X	*Rare*	*Random*

TABLE 4.: Taxonomy of some of the mesh-based P2P streaming protocols.

2.2.3.3. Case Studies

Table 2.2.3.2. presents a classification of the major mesh-based P2P content delivery protocols based on different aspects of the mesh-based approach. In this section, we describe some of the major proposed protocols in more details.

DoNet [64]

Early version of DoNet employs a mesh-based approach to support P2P streaming applications. DoNet requires each peer maintains a partial sub-set of other peers in the session, and participates in a continuously running gossiping algorithm (*e.g.*, SCAMP) for membership management. Newly joining peers contact a bootstrap node to obtain an initial set of potential partners and further, run the gossiping algorithm to discover new partners. DoNet constructs a bi-directional random mesh in which all peers have the same in- and out-degree regardless of their incoming or outgoing accesslink bandwidth.

The content delivery in DoNet is similar to PRIME, except that peers exchange the whole buffer map with their partners rather than newly available blocks. The scheduler is invoked periodically to determine the set of blocks that should be requested from each partner. DoNet employs a *Rare* block selection scheme. The scheduler determines the potential suppliers of blocks starting from those with only one potential supplier, then those with two, and so forth. Among the multiple potential suppliers, the one that has the lowest fraction of its bandwidth utilized is determined. Source in DoNet, does not perform any specific coordination.

Chainsaw [65]

Chainsaw organizes peers into a bi-directional random mesh in a centralized fashion. A new peer contacts a bootstrap node and obtains a set of neighbors. A peer attempts to maintain a specified minimum number of neighbors without considering its outgoing or incoming bandwidth. Peers never refuse a connection request from any peer.

Block scheduling in Chainsaw, is event-driven that is the scheduling scheme invokes whenever a neighbor's outstanding list of blocks is finished or a parent notifies of availability of a new block. This event-driven scheduling can potentially reduce the playout delay introduced by periodic scheduling, as each peer can request a block as soon as it becomes available at any parent. However, notifying neighbors whenever a new block becomes available and further, per-block individual requests incur a very high overhead. Chainsaw applies a *random* scheme to select blocks for requesting from neighbors, in case there exists more than one wanted blocks. The source has a specific behavior in Chainsaw, that the source maintains a list of blocks that have never been delivered before. When it receives a request for a block that is not on the list, the source ignores the requested block, sends the oldest block on the list instead. This behavior ensures that at least one copy of every block is delivered quickly, and the source's bandwidth does not utilize for sending duplicate copies of blocks.

Pulse [66]

Pulse is another mesh-based P2P protocol for streaming live content. It incorporates a mechanism similar to BitTorrent for rewarding peer participating and discouraging peers from contributing an insufficient amount of resources. It build a bi-directional random mesh in which all peers have the same in-degree while their out-degree is variable. In Pulse, peers do not have a synchronized playout time, therefore, peers can request blocks only from neighbors with overlapping buffer. The scheduling algorithm runs every constant period of time (called epoch) and it is based on an incentive mechanism similar to the one used in BitTorrent. In each scheduling event, a peer chooses a fixed number of potential parents from its neighbor set which upload most data to it during the last period and have overlapped buffer with the peer. Packets exchange with these parents can be mutual, if both sides have blocks the other one needs. If the peer has still available bandwidth, it will select more peers from its neighbor set by using a *history* parameter which indicates the quality of previous exchanges with other peers. These latter set of peers will be served with less priority and serving them introduces some altruism in the session, and allows high bandwidth peers to contribute more of their capacity to the session. The block selection schemed used by all peers is based on a rarest scheme and blocks are assigned to random peers.

2.2.4. Hybrid Approach

Hybrid approach is a combination of the above approaches and is suitable for one-to-many communication model. A hybrid approach can be achieved by combining push and pull for content delivery [22] or tree and mesh for the content delivery path [23]. In protocols incorporates mesh-tree hybrid approach, participating peers receive part of the content through a tree overlay and the rest of the content is pulled from mesh parents. Two existing protocols that incorporate the hybrid approach are Bullet [22] and mTreebone [23], which we further describe in detail.

2.2.4.1. Case Studies

Bullet [22]

Bullet is designed for elastic content delivery. In Bullet, peers are organized into an overlay tree, which can be constructed and maintained by any of the existing tree construction mechanisms [33, 18, 38, 58]. Each Bullet peer, starting with the source of the tree, transmits a *disjoint* set of blocks to each of its children, with the goal of maintaining uniform representativeness of the content across all participating peers. The level of disjointness is determined by the bandwidth available to each of its children. Bullet then employs a distributed peer discovery to enable peers to quickly locate multiple parents capable of transmitting missing blocks of the content to the peer without global knowledge. Bullet incorporates RanSub [72] to periodically disseminate the changing, uniformly random subsets of global state to each participating peer. Each peer uses this information to request missing blocks from other participating peers. RanSub distributes the content availability information of random subsets of participating peers through the tree using *collect* and *distribute* messages. Collect messages start at the leaves and propagate up the tree, leaving state at each peer along the path to the source. Distribute messages start at the source and travel down the tree, using the information left at the peers during the previous collect round to distribute uniformly random subsets to all participants. Each peer as a parent creates distribute sets for each child by compacting collects sets from that child's siblings and its own distribute set. The results is a state information of a random subset of participating peers representing all peers in the tree, except for those rooted at that particular child.

In essence, during each interval, a peer receives a summarized partial view of the session's state at that time. Upon receiving a random subset, a Bullet peer may choose a peer with the minimum amount of shared blocks compared to its own available blocks to be its perpendicular (mesh) parent. This is done only when the peer has sufficient slots based on its bounded in-degree (in Bullet it is 10) to request for another parent (parents with poor performance may be removed). Once a peer has chosen the potential perpendicular parent, it sends a peering request containing its Bloom filter. The new perpendicular parent will transmit blocks not present in the Bloom filter to the child peer. The child peer will refresh its Bloom filters at each of its perpendicular parents, periodically. Along with the fresh bloom filter, a child will also assign a pre-determined fixed portion of the sequence space to each of its perpendicular parents to reduce the likelihood of receiving duplicate blocks. A duplicate block, however, may be received when a parent (through the tree structure) recovers a block from one of its parents and relays the block to its children (and descendants). Less than 10% of all received blocks are duplicates in their experiments.

mTreebone [23]

mTreebone is another mesh-tree hybrid protocol which relies on constructing a tree with only stable peers as a backbone and pushing the streaming content over this backbone. These stable peers, together with other participating peers, are further organized a random mesh overlay, which facilitates accommodating churn and fluctuation of bandwidth in the backbone. The core of mTreebone is constructing a tree-based backbone, referred to as *treebone*. This backbone consists of only a subset of peers which are stable. Other non-stable peers are attached to the backbone as outskirts. The streaming content is pushed through the treebone and eventually reach the outskirts. To improve the resiliency and efficiency of the treebone, participating peers are organized into a mesh overlay. A gossip-based membership management algorithm is exploited for peers to periodically exchange their status. The mesh neighbors periodically exchange their content availability information. However, a peer is schedule to pull content from neighbors in the mesh only if an disruption in delivery of the content occurs in the treebone.

mTreebone identifies stable peers through their session length (peers with higher age tend to stay longer). If the session length of a peer exceeds a certain threshold, it will promote itself as a treebone peer and further can accept children. Upon joining, each peer obtains a partial list of participating peers from source, at least one of which is in the treebone. The new peer attaches itself to one of the treebone peers and locates mesh neighbors using the list. To optimize latency of the treebone, if a treebone peer has more children than a peer closer to the source in the treebone, a swap of them occurs to reduce the average depth of the treebone. Moreover, each treebone peer tries to move upward in the tree by periodically checking whether there are peers closer to the source than its parent that can accept a child. If so the peer leaves its original parent and attach itself to the closer peer as a child. Without any disruption in the delivery of the streaming content, peers keep receiving the whole content from their single parent in the treebone. In case of a departure of any treebone peers or bandwidth reduction in any upstream treebone connections, affected children try to request missing blocks from their mesh parents.

2.3. Issues & Challenges in Practical Deployment of Live P2P Streaming

Despite popularity of P2P content delivery applications, there are some issues that has not completely addressed yet and remained as open issues specially in the context of streaming content delivery. In this dissertation, we cover a subset of these issues and here we will provide the existing research studies on those topics, namely, incentive mechanism and resource management in P2P streaming systems, video encoding and P2P traffic localizations.

2.3.1. Incentive and Fairness

So far we have made an implicit assumption that peers can and are willing to cooperate in delivery of the streaming content. However, in P2P networks, this is not always the case. Several studies [73, 74] have shown that users of P2P networks tend to be selfish and try to benefit from the P2P network without contributing as much resources in return. An extreme example of uncooperative behavior is the "free-rider", where a peer only consumes the bandwidth without contributing any. Therefore, a proper incentive mechanism that encourages peers to contribute and upload as much as they can is critical in P2P streaming applications. In the absence of such a mechanism, the performance of the P2P streaming application can seriously degrades or it can be variable and unpredictable.

Providing incentives in highly dynamic P2P networks where it is difficult to identify peers and obtain the information about their past behavior in order to predict their future performance, can be a particularly challenging task. P2P file sharing applications adopted various incentive mechanisms [75, 76, 77, 78, 79, 80, 81], based on payment [76], or reciprocity, to encourage peers to contribute. However, designing incentive mechanisms for P2P streaming applications is more challenging than P2P file sharing applications due to the unique features of streaming applications *i.e.*, real-time constraints and bandwidth requirement. In general, incentive mechanisms can be divided into two categories of: payment-based, reciprocity-based methods.

2.3.1.1. Payment-based Mechanisms

Payment-based mechanisms force that the peer that receives bandwidth simply pays the parent for resources it consumes. [76] is one of the first studies that considered payment-based mechanisms in P2P file sharing applications. This study uses a game theoretical model to study the potential benefits of incorporating payment-based incentive mechanisms into P2P file-sharing applications. Various payment-based mechanisms have been proposed in the context of P2P file-sharing. However, these mechanisms seem highly impractical even in P2P file sharing applications, since they require an infrastructure for accounting and micro-payments.

2.3.1.2. Reciprocity-based Mechanisms

In reciprocity-based mechanisms, peers maintain histories of past behavior of other participating peers and use this information in their decision making process. The reciprocity mechanism can be based on *indirect reciprocity* or *direct reciprocity.* In indirect mechanisms, the decision of peer X about Y is based on the contribution of peer Y to the whole P2P network. In contrast, in direct-reciprocity mechanisms, peer X decides how to treat Y based only on the contribution of peer Y to X in the past. In general, indirect-reciprocity mechanisms are vulnerable to collusive behavior such as false accusation and false praise [82] that do not arise in direct-reciprocity mechanisms.

Based on the time duration needed for reciprocation, reciprocity-based mechanisms can be further divided into *reputation-based* and *instantaneous* mechanisms. Reputation-based mechanisms rely on the history of contribution of a peer to the P2P network [83, 84, 85]. Reputation of peers is proportional to their overall resource contribution, and peers with higher reputation are rewarded with better performance. [84] and [86] propose a reputation-based method where peers with higher reputation are awarded with preferential treatment in parent selection. In their proposed approach, reputation is accumulated over time across multiple streaming sessions. In general, reputation-based methods are suitable for asynchronous systems such as VoD and file-sharing applications where contribution and reward do not need to happen simultaneously and peers stay in the system long enough to build adequate reputation. In the context of live P2P streaming, empirical measurement studies have shown that the median session time of peers is very short (i.e. 25% of peers are in the system for less than 2 min) [87]. In such a dynamic system, the instantaneous contribution and demand have to be considered for a fair distribution of resources. We believe that designing an incentive mechanism that computes the instantaneous contribution of peers and allocates

Instantaneous direct reciprocity mechanisms relax the need for maintaining long-term state information, in the form of reputation. This simplifies the design and improves the robustness of the mechanism against collusive behavior. BitTorrent, a P2P file-sharing application, is a good example of direct reciprocity approach by adopting a tit-for-tat strategy [75]. In such an approach, peers upload to peers from whom they are able to download at a higher. Studies found much lower levels of free-riding in BitTorrent network compared to other P2P file sharing applications. However, measurements and analysis has demonstrated that the BitTorrent protocol can still be manipulated by misbehaved peers in their favor and the fairness properties of BitTorrent is questionable [88, 89]. Although, the tit-for-tat strategy or its extended versions [90, 91] work well in the context of file sharing, it cannot be trivially extended to the context of P2P streaming because of the timeliness and the high bandwidth requirements involved. Moreover, the direct-reciprocity incentive mechanism requires direct interactions between each pair of peers which might have some implications on the properties of the overlay structure for P2P streaming applications.

In the context of live P2P streaming, [92] proposes an extension of BitTorrent's tit-for-tat strategy for parent selection based on local information of available bandwidth and streaming content among neighbors. Similarly, [93] and [94] extending the tit-for-tat strategy, leverage the layered encoded streaming to accommodate heterogeneity of bandwidth and enable video quality adaptation. As such direct reciprocity approaches focus on peers local information, the aggregate excess resources are randomly distributed (instead of proportionally) among peers depending on their neighbors.

Another flavor of instantaneous indirect reciprocity mechanism is to encourage peers to contribute as much bandwidth as they have rather than focusing on a bit-for-bit fair sharing where a peer receives as much bandwidth as it contributes. This mechanism addresses the resource management issue in the context of live mesh-based

P2P streaming over a wide range of scenarios such as highly heterogeneous and asymmetric peers bandwidth and realistic churn model. For that, authors in [95, 96], consider taxation schemes in which bandwidth-rich peers try to compensate the resource-poor peers by contributing more bandwidth to the system and proportionally receiving higher bandwidth. The intuition is that with a bit-for-bit mechanism, peers with very high upload capacity (*i.e.*, behind Ethernet) that are capable of contributing much more than the source rate, can not contribute all of their resources, while peers with low upload capacity (*i.e.*, behind asymmetric connections such as DSL and cable modem) are precluded from receiving more than their upload capacities. The basic mechanism in this approach, is a contribution-aware framework where peers receive different levels of bandwidth based on the overall instantaneous upload bandwidth available in the system as well as the amount of resources the peer contributes. This contribution-aware mechanism is of a flavor of indirect-reciprocity mechanisms and relies on trusting the participating peers about their announced instantaneous contributions to the system.

2.3.2. Encoding

In a P2P network, typically, peers have various incoming access link bandwidth (*e.g.*, peers behind Ethernet, dial-up and DSL) and the paths between peers might have different capacity. Moreover, in P2P streaming applications, the rate at which peers receive the streaming content might fluctuate due to departure of a parent[97] or congestion at the core of the network. A P2P streaming application should be able to accommodate both heterogeneity of bandwidth and long-term variation of available bandwidth, and offer quality of the stream proportional to each peer's bandwidth, *i.e.*, low and high bandwidth peers should be able to receive low and high streaming quality, respectively.

To address this issue, some studies proposed maintaining multiple streams with various qualities at the source [98, 99, 100]. Each stream is delivered into a separate stream channel. Peers can join to a particular channel based on their incoming available bandwidth and switch between channels accordingly. The streams are encoded with different compression parameters. Having multiple streaming channels leads to wasting the network resources (*i.e.*, bandwidth) as a single stream is duplicated multiple times.

Another approach is incorporating coding algorithms into video/audio. In essence, streaming content can be encoded with coding techniques such as *Layered Coding (LC)* [101, 102] or *Multiple Description Coding (MDC)* [103, 104].

In LC, the streaming content is encoded in a base layer and one or more additional layers. While, the base layer can be independently decoded, the additional layers can be decoded cumulatively *i.e.*, layer m can be only decoded along with lower layers of 1 to $m - 1$. Additional layers improve the quality of the video content. Incorporating LC, allows the peers to add/drop sequential layers to adapt their quality based on their aggregate incoming bandwidth [105, 106, 107]. P2P streaming applications that employ LC are sensitive to losses of lower layer packets which are more important.

MDC is used in various protocols proposed for live P2P streaming[56, 29, 4]. MDC is a method of encoding an audio/video content into several separate layers/descriptions with some redundancy between them. The layers/descriptions are independent and any subset of them can be decoded and merged to improve the quality. However, this easiness, can be achieved at the expense of a slightly higher stream rate than the original stream rate without the encoding.

Some studies have investigated the impact of LC and MDC on the performance of P2P streaming applications. In [108], authors study the effect of LC and MDC on the performance of on-demand P2P streaming applications. Authors, suggest that the design of a P2P streaming application is simpler with MDC than LC, but the required bandwidth for delivery of MDC encoded stream is higher than an LC encoded stream. In essence, LC allows the peers to accommodate more requests for the same streaming quality. On the other hand, MDC is simpler, gives higher flexibility and resiliency to losses.

2.3.3. Traffic Localization

Traffic generated by P2P applications makes up a substantial fraction of today's Internet traffic. In P2P applications, participating peers often form an overlay which is largely agnostic to the underlying physical topology [109, 110]. This in turn increases the cost associated with P2P traffic for individual Internet Service Providers (ISPs) which is a serious concern. This problem has motivated the idea of localizing P2P traffic within individual ISPs by localizing the connectivity among their peers [111, 112]. The common assumption in this approach is that localizing the overlay connectivity has minimal impact on the performance of P2P applications. Thus, the main focus on these studies is on providing an interface between ISPs and P2P applications to facilitate the localization (*i.e.*, identifying local peers) [111, 113] A few studies examined the performance of file swarming mechanisms over localized overlay [114, 111].

The existing works that tackle the problem of P2P traffic localization can be divided into three classes: *(i)* the ones with the goal of building a localized P2P overlay, *(ii)* those which investigate the effect of P2P traffic localization on the performance of P2P file sharing applications, and *(iii)* recent studies that focused on localization in live P2P streaming systems.

Regarding building a localized overlay, several studies have proposed using IP-to-AS mapping tools [115, 116]. Recently, [112] has suggested cooperation between ISPs and P2P applications to control the Inter-ISP traffic. The proposed solution require an oracle to provide information about the location of peers. [114] has proposed that the information about location of peers can be provided by ISPs to P2P applications which might have security issues. P4P [111] project has proposed deploying an interface between ISPs and P2P applications to solve the trust issues and simplify applying any policy-based peer selection. A recent study [113] has proposed technique for finding close-by peers that requires no cooperation or trust between ISPs and the application, no infrastructure information. This lightweight scalable peer selection technique relies on the information collected by CDN servers. While the above solutions can be integrated for peer discovery in any ISP-friendly P2P application, these works either do not directly investigate the effect localization on the performance of P2P applications or they focus on the performance of P2P file sharing applications *i.e.*, BitTorrent.

[114] has focused on effect of traffic localization on BitTorrent performance and tried to identify scenarios that localization does not reduce performance such as enough number of seeds in each ISP, population of ISPs and contribution of peers inside ISPs. [111] shows an improvement in the performance of localized file sharing applications due to a potentially higher available bandwidth within an ISP.

The above studies focus on file sharing applications, however, in-time delivery of packets and limited availability of content impose unique requirements for scalable live P2P streaming applications. [117, 118] recently proposed a scheme for ISP-friendly mesh-based live streaming. Their solution builds a clustered primary overlay augmented by a large number of dynamically-unchoked, secondary inter-cluster links. Their goal is to use internal connections for receiving the content and use external connections in a demand-driven fashion, when there is not an adequate amount of new content among internal neighbors. These solutions do not guarantee a lower bound on the amount of costly traffic between ISPs.

CHAPTER III

PRIME: PEER-TO-PEER RECEIVER-DRIVEN MESH-BASED STREAMING

Material in this chapter was adapted from papers previously published in a journal, conferences, a workshop and as a poster [8, 4, 119, 3, 2, 1]. The papers was co-authored with Prof. Reza Rejaie and Dr. Yang Guo. The poster was co-authored with Prof. Reza Rejaie, Dr. Danial Stutzbach and Amir H. Rasti. The experimental work is entirely mine. The text is mostly mine with some contributions from Prof. Reza Rejaie and Dr. Yang Guo who also provided technical guidance.

During recent years, the increasing ability of average users to generate multimedia content coupled with the availability of high bandwidth connections especially to residential users, motivated research on streaming multimedia content over the Internet. A popular streaming application is one-to-many streaming of *live* video over the Internet (*e.g.*, IPTV [120]). Peer-to-Peer (P2P) overlays offer a promising approach to one-to-many streaming of live video over the Internet without any special support from the network. This approach is often called *P2P streaming.* In P2P streaming, participating peers form an overlay and contribute their outgoing bandwidth by sending the streaming content to other peers. The goal of P2P streaming mechanisms is to deliver high quality stream to individual peers in a scalable fashion. To gracefully scale with the number of participating peers in a session, a P2P streaming mechanism should be able to effectively utilize the contributed resources (*i.e.*, outgoing bandwidth) of individual peers. However, achieving scalability despite heterogeneity and asymmetry of access link bandwidth among peers and their dynamic participation is challenging.

Most of the existing P2P streaming applications form a tree-shaped overlay where the content is pushed through the overlay, from a source (*i.e.*, root) towards all peers. This approach cannot provide high quality stream to participating peers due to the following reasons: *(i)* It can not utilize outgoing bandwidth of all participating peers (particularly leaves of the tree). *(ii)* Delivered quality to each peer is limited to the minimum bandwidth among the upstream connections from source. *(iii)* Departure of individual peers could disrupt the delivered quality to its down stream peers. *(iv)* Maintaining an efficient tree is expensive due to the dynamics of peer participation. An extended version of this approach organizes peers into multiple trees where each peer is an internal node in only one tree and a leaf node in all other trees [29]. Then individual descriptions of a multiple description coded (MDC) stream is pushed through each tree. This approach improves utilization of resources and resiliency to peer dynamics, however due to the static mapping of content to trees, delivered quality to participating peers is still limited [4].

The limitations of tree-based approaches have motivated a new approach which we call *mesh-based* approach in which participating peers form a random mesh and incorporate *swarming* (or pull) content delivery. The mesh-based approach has been motivated by the success of file swarming mechanisms such as BitTorrent. File swarming mechanisms (e.g. Bittorrent) leverage the elastic nature of the content and the availability of the entire file at the source to effectively utilize available resources and scale. More specifically, in file-swarming mechanisms, source distributes different pieces of a file among peers which enables them to contribute their out-going bandwidth more effectively. Individual peers pull different pieces of the file in a random or rarest-first order and potentially at different rates from their neighbors in the overlay.

3.1. Contributions & Design Objectives

Incorporating swarming content delivery into mesh-based P2P streaming mechanisms for live content is challenging for two reasons: *(i)* Swarming does not guarantee the in-time delivery of individual blocks of streaming content to peers. *(ii)* Since the content is progressively generated by the live source, the amount of new blocks is limited. This reduces the diversity of available pieces among participating peers which in turn degrades the utilization of their outgoing bandwidth and leads to ineffective swarming.

Despite challenges in incorporating swarming content delivery into live P2P streaming, a couple of recent studies showed that it is feasible to incorporate swarming into P2P streaming in certain scenarios [64, 65, 121]. However, to our knowledge, none of the previous studies have answered the following important questions:

- How can the swarming content delivery be incorporated into a live mesh-based P2P streaming mechanism to effectively scale with the peer population?
- What are the fundamental tradeoffs and limitations in design of such a scalable mesh-based P2P streaming mechanism for live content?

The first contribution of this chapter is to address these two important questions. Through a performance-driven approach, we design PRIME, a novel mesh-based P2P streaming mechanism for delivery of live content. Initially we identify two performance bottlenecks in mesh-based P2P streaming that could limit the utilization of available resources and thus limit its scalability as follows: *(i)* A peer experiences *bandwidth bottleneck* when its aggregate rate of content delivery from all of its neighbors is not sufficient to fully utilize its incoming access link bandwidth. Further, we show that the probability of bandwidth bottleneck directly depends on the connectivity of the overlay (*i.e.*, the incoming and outgoing degrees of individual peers). We then derive the proper connectivity for individual peers that minimizes the probability of bandwidth bottleneck among them. *(ii)* A peer experiences *content bottleneck* when there is not sufficient amount of useful content among its neighbors to effectively utilize its available bandwidth. We show that the probability of content bottleneck among peers directly depends on the global pattern of content delivery from source to all peers in the overlay. We introduce the “organized view” of a random mesh and then derive the desired pattern of content delivery for a single segment that minimizes the probability of content bottleneck among peers and thus maximizes the utilization of resources to accommodate scalability. We demonstrate that the desired pattern of delivery should consist of two phases: *(i)* a *diffusion* phase where data rapidly flows away from source, and is followed by *(ii)* a *swarming* phase where peers exchange their available blocks. We derive the required “block-pulling” strategy at individual peers that its collective behavior across all peers leads to the desired pattern of delivery. The two-phase view of the content delivery leads to two important insights: *(i)* It reveals the impact of overlay connectivity and source behavior on the performance of content delivery. *(ii)* It demonstrates some fundamental limitations of the system by illustrating the relation between peer population, overlay connectivity and minimum buffer requirement at individual peers.

The second contribution of this chapter is the detailed performance evaluation of PRIME using block level simulations. We show that the notion of diffusion and swarming phases offers a powerful method to identify the performance bottlenecks of a mesh-based P2P streaming mechanism. We carefully examine the performance of PRIME in scenarios with limited resources and untangle the effect of different parameters on overall performance of PRIME. Our results not only reveal a few fundamental design tradeoffs and limitations in incorporating swarming content delivery into mesh-based P2P streaming for live content but also shed an insightful light on the dynamics of swarming content delivery in these systems.

3.2. Existing Mesh-based Solutions

In recent years, streaming media over P2P overlays has received significant attention and many studies have been done on this topic. In this section, we focus on a few previous studies that are most related to our work.

CoolStreaming/DONet [64] is a mesh-based approach where peers initially form a mesh [122]. However, once each peer identifies proper parents, it requests each parent to provide a specific sub-stream of the content. In essence, CoolStreaming eventually organizes peers into multiple trees and incorporates push-based content delivery [122]. Using prototype implementation, authors conduct experiment over PlanetLab and report on their experience with large scale deployment of this system. Authors present average delivered quality to the peers as a function of peer degrees (over a small range from 2 to 6) and churn. While this study clearly demonstrates the scalability of mesh-based P2P streaming, it does not demonstrate the fundamental tradeoffs in the design of mesh-based P2P streaming mechanisms.

Several studies have proposed to add the notion of "delivery window" to Bittorrent in order to support "streaming" content delivery (*e.g.*, [65, 68, 123]). These studies appear to be targeting playback streaming or on-demand applications. One important difference between live and on-demand P2P streaming is the availability of content for swarming content delivery. In VoD applications the entire content is usually available which increases the diversity of available content among peers and accommodates swarming content delivery. However, in the context of live P2P streaming applications such as PRIME, accommodating swarming content delivery is more challenging because the useful content for swarming is being gradually generated by the source and is more limited. Therefore, the performance of the proposed on-demand P2P streaming mechanisms with limited available content and limited resources is unknown. Finally, a growing number of P2P streaming systems (*e.g.*, wwitv.com, sopcast.com) have become available for broadcasting the streaming content to a large group of end-systems over the Internet. However, no technical details about these systems is available for comparison.

3.3. Overlay Construction in PRIME

Participating peers in PRIME, form a randomly connected and directed mesh. All connections are uni-directional *i.e.*, there is a parent-child relationship between connected peers. Each peer, has multiple parents and multiple children. Each peer as a child, identifies sufficient number of peers as parents. To discover potential parents, individual peers contact a bootstrapping node to learn about other existing peers in a demand-driven fashion. Such an overlay has several advantages: *(i)* Building and maintaining the overlay is simple and has low overhead, *(ii)* it is resilient to churn and *(iii)* connections are more diverse thus it is less likely that incoming connections from

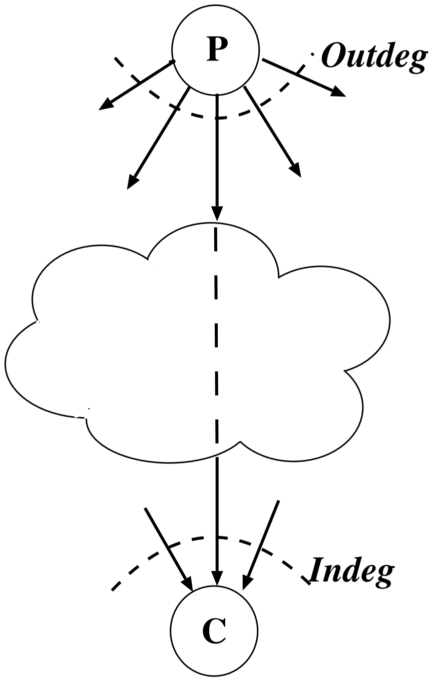


FIGURE 3.: A connection from peer p to peer c with outgoing and incoming access link bandwidth of $outbw_p$ and $inbw_c$, respectively.

parents to a child share a bottleneck inside the network. We note that PRIME can certainly incorporate other (distributed or central) peer discovery and parent selection techniques. However, as long as the incoming and outgoing degrees of individual peers are not affected, other details of these techniques do not have any significant impact on PRIME performance.

To construct the overlay, each peer tries to maintain a sufficient number of parents that can collectively fill its incoming access link bandwidth. All connections in the overlay are congestion controlled (using RAP[124] or TFRC[125]). The key design question for the overlay construction mechanism is *"how to determine the incoming and outgoing degree of individual peers?"*

3.3.1. Bandwidth-Degree Condition

Suppose that each peer always has sufficient amount of useful content to send to its children. Then the aggregate rate of content delivery to each child peer depends on *(i)* the number of its parent peers (*i.e.*, indegree) and *(ii)* the number of child peers for those parents (*i.e.*, parents' outdegree). *i.e.*, at the incoming or outgoing access links of participating peers. Figure 3. shows a child peer with incoming access link bandwidth of $inbw_c$ and indegree of $indeg_c$ as well as one of its parent peer with outgoing access link bandwidth of $outbw_p$ and outdegree of $outdeg_p$. Suppose that congestion occurs only at the edge of the network, *i.e.*, the incoming/outgoing access links of participating peers. Therefore, the average bandwidth of the connection between the child peer c and the parent peer p in Figure 3., can be estimated by $MIN(\frac{outbw_p}{outdeg_p}, \frac{inbw_c}{indeg_c})$. If $\frac{outbw_p}{outdeg_p} < \frac{inbw_c}{indeg_c}$, the outgoing access link of parent p is the bottleneck and the incoming access link of child c can not be fully utilized. On the other hand, if if $\frac{outbw_p}{outdeg_p} > \frac{inbw_c}{indeg_c}$, the incoming access link of child c is the bottleneck and the outgoing access link of parent p may not be fully utilized.

This observation suggests that to minimize the bandwidth bottleneck in a randomly connected mesh-based overlay, the same bandwidth to degree ratio should be used for all connections of all participating peers. We call this the $bandwidth-degree$ condition which should be satisfied by all participating peers i and j

$$bwpf = \frac{outbw_i}{outdeg_i} = \frac{inbw_j}{indeg_j}.$$

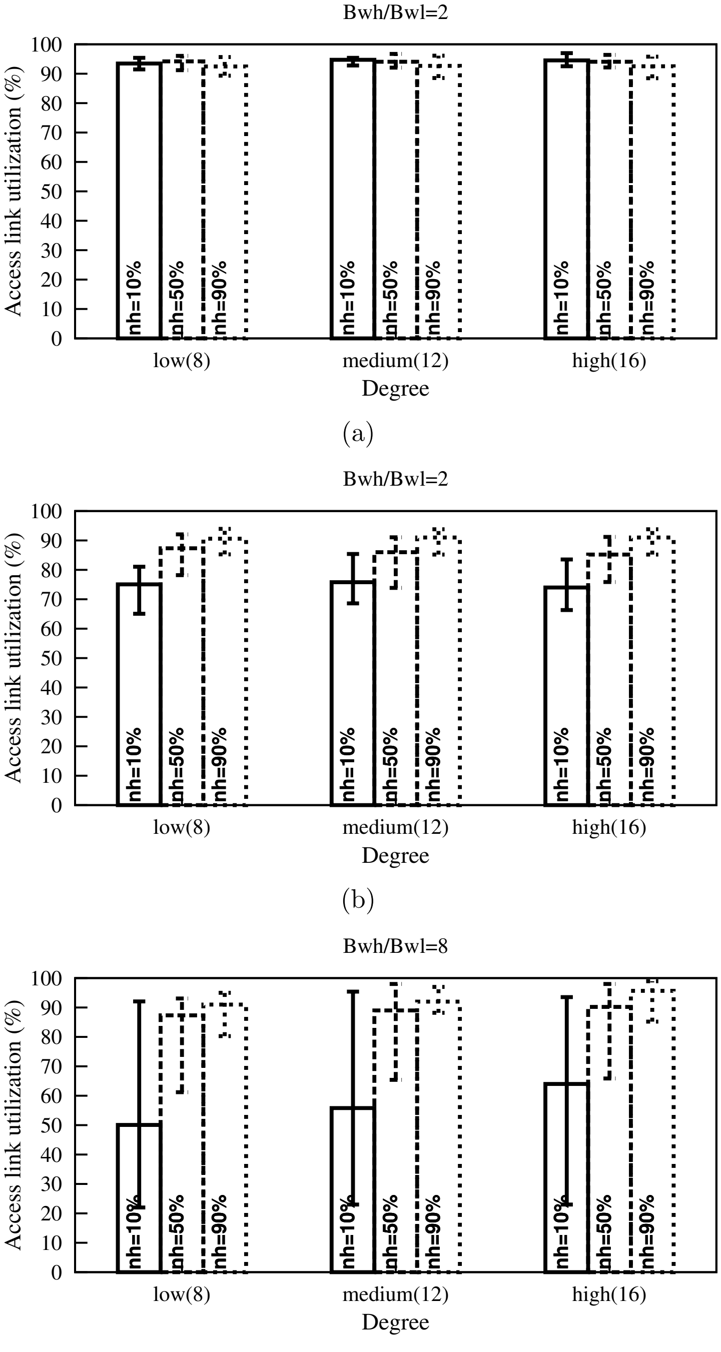
This condition implies that all connections in the overlay should have the same bandwidth of $bwpf$, or bandwidth-per-flow. Note that this can easily accommodate heterogeneity of bandwidth among peers by choosing a proper in/out degree to have the same $bwpf$ across all connections. In essence, $bwpf$ is a configuration parameter that directly translates the (potentially heterogeneous and asymmetric) access link bandwidth of individual peers (and the source) to their proper incoming and outgoing degree.

To illustrate the effect of bandwidth-degree condition on the utilization of access link bandwidth, we conduct *ns* simulations where 200 peers with symmetrical access link bandwidth of bw_h or bw_l form a directed and randomly connected mesh. We examine the performance with fixed degree across all peers as 8, 12 and 16. In addition to demonstrate the effect of $bandwidth-degree$ condition, we set the degree of peers proportional to their access link bandwidth in another sets of simulations. We keep the total number of connections fixed in all simulations for each degree for a fair comparison. Connections are congestion controlled using RAP[124]. We examine the performance with two different ratios of bandwidth heterogeneity *i.e.*, $\frac{bw_h}{bw_l}$ (2 and 8) among peers while keeping the low bandwidth fixed as 700 Kbps. We also change the percentage of high bandwidth nodes (nh) between 10%, 50% and 90%, to examine its impact on utilization of access link bandwidth. Figures 4.(a), 4.(b) and 4.(c) show the average utilization of incoming access link bandwidth and its 10th and 90th percentiles (as bar) only among high bandwidth peers (bw_h) for 3 scenarios: *(a)* $bandwidth-degree$ condition, *(b)* fixed degree with $\frac{bw_h}{bw_l} = 2$ and *(c)* fixed degree with

$\frac{bw_h}{bw_l} = 8$, respectively. Each figure shows bandwidth utilization in three different peer degree settings *i.e.*, low, medium and high. Within each degree setting, the three boxes show access link utilization for three bandwidth settings: 10%, 50% and 90% of the population being high bandwidth.

These figures illustrate the following points: *(i) Without $bandwidth - degree$ condition, utilization of access link among high bandwidth peers is not full and it has high variations.* The average access link utilization in scenario with enforced $bandwidth - degree$ condition is always more that 95% with low variation ($< 3\%$). *(ii)* If all peers use the same degree, increasing the degree of bandwidth heterogeneity decreases the average utilization of access link bandwidth among high bandwidth peers especially when the fraction of high bandwidth peers is small (*e.g.*, $\frac{bw_h}{bw_l} = 8$ and n_h=10%). *(iii)* Increasing percentage of high bandwidth peers (*i.e.*, $nh = 10\%$, 50% vs. 90%), improves their access link bandwidth utilization due to the increasing number of connections among them. *(iv)* Similarly increasing peer degree results in higher utilization of access link among high bandwidth peers due to the larger number of connections among them. In summary, accommodating the bandwidth-degree condition ensures that each peer can receive content at the maximum rate and does not experience a *bandwidth bottleneck*.

In practice, the observed bandwidth for congestion controlled connections in the overlay is likely to be different due to the difference in their round-trip-time or loss rate. Furthermore, some connections might experience bottleneck in the core



(a)

(b)

(c)

FIGURE 4.: Effect of bw-degree condition: Utilization of access link bandwidth across different peer degrees and levels of heterogeneity. (a) $bandwidth - degree$ condition is enforced. (b) and (c) Fixed degree with bandwidth heterogeneity ratio of 2 and 8, respectively.

rather than the edge of the network. This may affect the utilization of access link bandwidth for the children that receive content through these connections. This problem can be addressed by incorporating an adaptation scheme that *(i)* allows children with low utilization of incoming access link bandwidth to have extra parents and *(ii)* allows parents with poor utilization of outgoing access link bandwidth to accept extra children beyond the limit that is specified by the bandwidth-degree condition. We note that the above adaptation scheme should be used for minor tuning of incoming/outgoing peer degree and can not replace the bandwidth-degree condition. Given the dependency of congestion control bandwidth of individual connections to the degree of corresponding peers, the degree of each peer affects not only its own bandwidth utilization but also the bandwidth utilization of its children or parent peers. If peers independently try to determine their proper incoming/outgoing degree, the ripple effect of this decision could easily lead to instability of the overlay. The bandwidth-degree condition provides an implicit coordination for individual peers to determine their degree in a coherent fashion and thus avoids any oscillations in the overlay.

3.4. Content Delivery in PRIME

Similar to other swarming mechanism, content delivery in PRIME incorporates push reporting coupled with pull requesting. Each peer simultaneously receives content from *all* of its parents and provides content to *all* of its children. Each peer, as a parent, progressively piggybacks a list of newly available blocks to its child peers within each data blocks. Given the available blocks at individual parents, a *block scheduling* scheme at each peer periodically (*i.e.*, once per Δ second) determines an ordered list of blocks that should be requested from each parent. Parent peers deliver requested blocks in the provided order and at the rate that is determined by the congestion control mechanism. To accommodate bandwidth heterogeneity among peers, the stream is encoded using a Multiple Description Coding (MDC) scheme at source. [1]

In the context of live P2P streaming applications, source progressively generates a new segment of content once every Δ seconds where a segment consists of a group of blocks with consecutive timestamps ([t_{src}-Δ,t_{src}]) across all descriptions, and t_{src} denotes source's playout time. To effectively accommodate swarming, peers should maintain a loosely synchronized playout time which is ω*Δ seconds behind source's playout time. Maintaining synchronized playout time maximizes the overlap among buffered data at different peers by providing roughly ω*Δ seconds worth of content that can be swarmed among peers. This also facilitates parent selection because each peer with open slot can serve as a parent[2]. The relative playout delay between the source and peers has two implications: *(i)* each peer should buffer at least ω*Δ

[1] In MDC coding, there is no decoding dependency among descriptions. Therefore any subset of descriptions can be decoded (viewed) by each peer.

[2] While this may seem intuitive, some of the P2P streaming mechanisms [66] have assumed that a peer has to delay its playout compare to its parents to provide more time for content delivery. This approach could lead to a long delay between source and some peers, and would limit the choices of parents to only those peers that have earlier playout time.

seconds worth of content, and *(ii)* each block should be delivered within ω*Δ seconds from its generation time to ensure in-time delivery.

Avoiding Content Bottleneck: Suppose all connections have roughly the same bandwidth ($bwpf$), then the maximum amount of data that a child can receive from a parent during an interval (Δ) is equal to $D = bwpf{*}\Delta$. This amount of data is called a *data unit* and consist of several blocks (possibly from different descriptions) that are selected by the block scheduling scheme at a child. When one (or multiple) parent(s) of a child do not have a data unit worth of new content to deliver during an interval, the child cannot fully utilize the bandwidth from the corresponding connection(s) and experiences *content bottleneck*.

3.4.1. Global Pattern of Content Delivery

The goal of the block scheduling scheme at individual peers is to maximize their delivered quality with minimum buffer requirement. This goal can be achieved by minimizing the probability of content bottleneck among peers which in turn maximizes the utilization of the outgoing bandwidth among all peers and thus improves scalability. The probability of content bottleneck among peers (*i.e.*, the availability of new data units at individual parents) directly depends on the global pattern of content delivery from the source to all peers through the overlay. Therefore, to design a scalable P2P streaming mechanism, first we identify the global pattern of content delivery that minimizes the probability of content bottleneck among peers.

Then, we derive the required block scheduling scheme at individual peers that leads to the desired global pattern. We describe the global pattern of content delivery for a single segment of content. Consecutive segments of the stream can be pipelined through the overlay by sequentially following a roughly similar pattern.

3.4.2. Organized View of a Random Mesh

To identify the desired global pattern of content delivery, first we present an organized view of a randomly connected and directed mesh. Toward this end, we group peers into levels based on their shortest distance from source. Peers that are exactly one hop away from source, are grouped into level 1, peers that are two hops away from source are located in level 2 and so on (as shown in Figure 5.). This view of the overlay reveals some simple but important properties of a mesh-based overlay. Consider the overlay that consists of P homogeneous peers with the same in- and out-degree deg and source degree of deg_{src}. Such an overlay exhibits the following properties:

- The population of peers at level i ($pop(i)$) is at most $deg_{src}{*}deg^{(i-1)}$ and simply this reveals that by going down through the levels populations of increases.
- The number of levels ($depth$) of the overlay can be estimated as $log_{deg}(P/deg_{src})$ $\leq depth$.

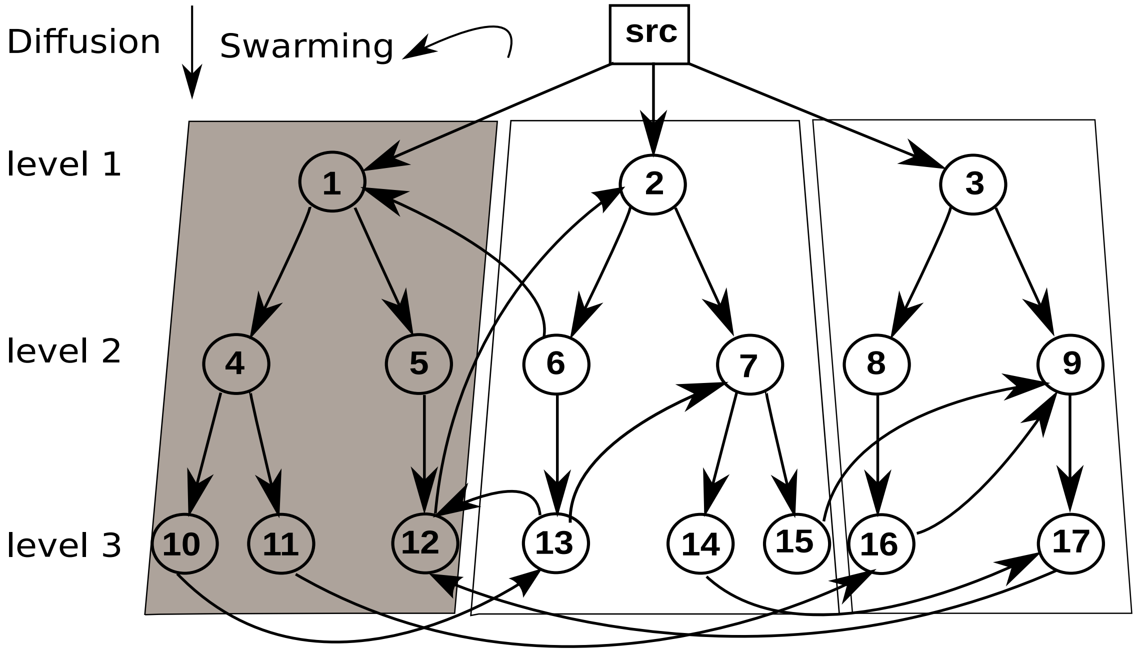


FIGURE 5.: Organized view of a random mesh overlay with 17 peers, forming 3 diffusion subtrees. For clarity, only a subset of connections are shown.

- The probability of having a parent at level i is equal to $\frac{pop(i)}{P}$. Each peer in level i, typically has a single parent in level $i-1$ which we call diffusion parent, and $deg-1$ parents in the same or lower levels which we call swarming parents. In practice, small number of peers may have more than one parents in the higher level due to the random overlay construction, this reduces the populations of peers in their corresponding level and might increases the *depth* of the overlay.

3.4.3. Pattern of Delivery for a Single Segment

Our goal is to derive the global pattern of content delivery for a single segment of content that minimizes the probability of content bottleneck among peers. Consecutive segments of the stream can be delivered through the overlay using a roughly similar pattern. Intuitively, to minimize the number of intervals for delivery of a segment, first different data units of the segment should be rapidly delivered (or diffused) to different subset of peers. Then, peers can exchange (or swarm) their data units and contribute their outgoing bandwidth until each peer has a proper number of data units for that segment. This observation motivates a two-phase approach for the delivery of a segment as follows: In the first phase, once a segment is generated at the source, all participating peers should receive a data unit of that segment as fast as possible (*i.e.*, diffuse the segment to all peers). In the second phase, peers can exchange (*i.e.*, swarm) their data units with each other to receive a number of data units for that segment corresponding to their desired quality. In a nutshell, content delivery for a segment occurs at 2 phases of *Diffusion* and *Swarming*. Next, we will describe these two phases in more details.

3.4.3.1. Diffusion Phase of Delivery

Upon generation of the segment in the source, all peers in level 1 should collectively pull all data units of that segment from source during the next interval Δ, this is the start of diffusion phase for this segment. Peers in level 2 at the next interval Δ should pull those data units from their *diffusion parents* in level 1 and so on. Therefore the fastest time for delivery of all different data units of the segment to the lowest level *depth* is $depth{*}\Delta$ seconds. We call this *diffusion time* of a segment.

To rapidly diffuse a new segment to peers in lower levels of the overlay, all connections from diffusion parents should be exclusively used for diffusion of new data units. These connections are shown by straight lines in Figure 5. and are called *diffusion connections.* The number of diffusion connections between each two levels is at least equal to the number of peers in lower level (*i.e.*, $deg_{src}{}^{*}deg^{(n-1)}$) which is exponentially increasing with n[3].

By explicitly using diffusion connections for diffusion of new data units, after *depth* intervals each participating peer in the overlay has one data unit of the segment within $depth{*}\Delta$ seconds from its generation time. This restriction has the following implications: *(i)* Diffusion phase for a segment takes *depth* intervals. *(ii)* each peer p in level 1 as well as all the peers in a subtree rooted at p receive the same data unit of the segment during the diffusion phase of that segment but at different intervals based on their level. We call these subtrees, *diffusion subtrees.* Figure 5. shows the diffusion subtree rooted at peer 1. *(iii)* The number of diffusion subtrees is equal to the deg_{src}, but the uniqueness of data units in each subtree depends on source bandwidth that may cause redundancy between subtrees. *(iv)* When the bandwidth of a diffusion connection decreases to less than $bwpf$, all the downstream peers in that diffusion

[3] Note that, due to the possibility of having multiple diffusion parents number of diffusion connections might be more that number of peers in lower level

subtree experience content bottleneck during the diffusion phase. *We emphasize that the diffusion sub-trees are implicitly formed as a result of pull block scheduling by individual peers. Therefore, the diffusion sub-trees are* **not** *explicitly formed among peers and Thus, diffusion sub-trees do not need to be explicitly maintained.*

3.4.3.2. Swarming Phase of Delivery

At the end of the diffusion phase of a segment all participating peers have one data unit of that segment. During the swarming phase, each peer should pull missing data units of the segment from its swarming parents. The number of unique data units that each peer should receive for each segment depends on its required quality (which is proportional to its available bandwidth). The connections from swarming parents are called swarming connections and are exclusively utilized for swarming. These connections are shown with curly arrows in Figure 5.. As we have mentioned previously, all peers on a particular diffusion subtree receives the same data unit of a segment during its diffusion phase. Therefore only a swarming parent that is located at a different diffusion subtree can rapidly provide a useful data unit of that segment to a child peer. For example, in Figure 5., p_9 can immediately obtain a new data unit from p_{15} but not from p_{16}.

To receive its maximum deliverable quality, each peer with in-degree *deg* should receive *deg* different data units. Ideally, if all swarming parents of a child peer located at $deg-1$ different diffusion subtrees, the child peer can pull $deg-1$

unique data units in the first interval of swarming phase (*e.g.*, p_{12} in Figure 5.). Due to the random connectivity among peers, some swarming parents of each peer may reside on the same diffusion subtree and this causes content bottleneck in swarming phase for each peer (*e.g.*, p_9 in Figure 5.). However, during extra swarming intervals some of these swarming parents (on the same or different subtrees) will obtain new data units of the segment and can provide them to the child peer. For example, p_{16} receives a new data unit from p_{11} after one interval and can pass it to p_9 in the next interval. This implies that the minimum number of required intervals to receive $deg{-}1$ unique data units of a segment (swarming phase) may be more than one for each peer, depending on the location of its swarming parents.

In a randomly connected overlay, the probability of experiencing a content bottleneck during the swarming phase depends on the relative value of peer's incoming degree and the number of diffusion sub-trees with a unique data unit as well as the population of peers in the bottom level of each diffusion sub-trees. For a given overlay, the minimum number of swarming intervals (or K_{min}) is determined such that a majority of peers can receive their required number of data units (*i.e.*, proper number descriptions) of a segment. In Section 3.6., we show how the value of K_{min} is affected by other system parameters. In summary, the required buffer at individual peers or their relative playout delay compare to source (*i.e.*, $\omega*\Delta$ seconds) should be sufficiently long to accommodate both diffusion and swarming intervals for almost all peers by satisfying the following condition: $(depth{+}K_{min}){\leq}\omega$.

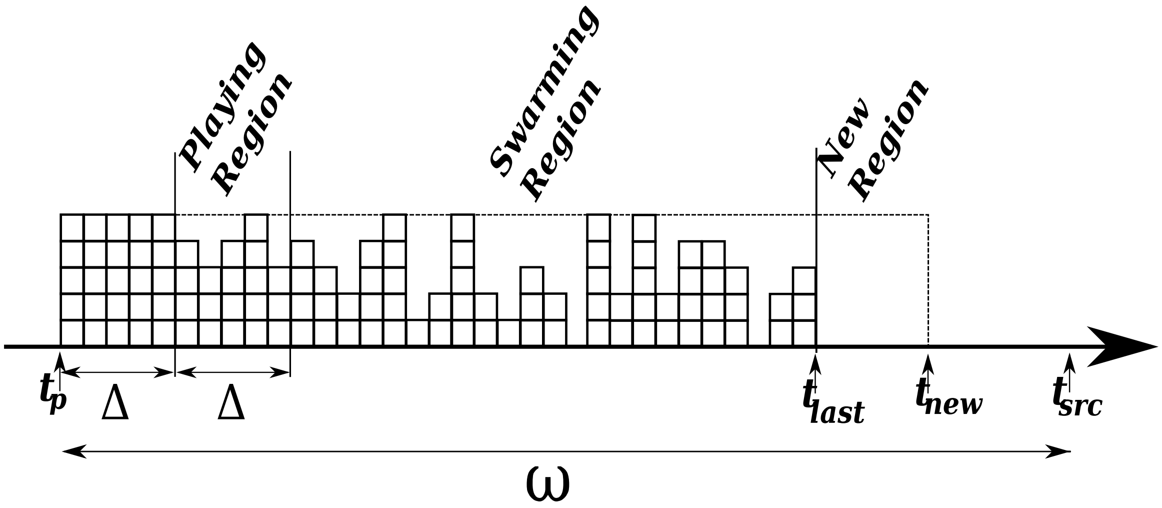


FIGURE 6.: Buffer state at a scheduling event in a peer.

3.4.4. Receiver-driven Block Scheduling

The block scheduling scheme at each peer determines requested (*i.e.*, pulled) blocks from individual parents. We assume that each block can be uniquely identified by its description id and a timestamp. The block scheduling at each peer takes the following input parameters: *(i)* the peer's target quality (*i.e.*, number of descriptions) that are being played (n), *(ii)* the exponentially weighted moving average of congestion controlled bandwidth from each parent ($ewma\_bw(i)$), *(iii)* reported blocks by individual parents that are required, and *(iv)* peer's own playout time (t_p) as well as the blocks that it has already received (*i.e.*, its buffer state). Given the above information, the block scheduling scheme at each peer should determine requested blocks from each parent in order to maximize the utilization of their available bandwidth.

Block scheduling at individual peers should behave such that its collective behavior leads to the desired pattern of content delivery. This in turn minimizes content bottleneck among peers. To achieve this goal, we divide the relevant blocks at each scheduling event into the following regions based on their timestamps as shown in Figure 6.. In Figure 6., t_p is the playout time and increases by stream rate, t_{max} is the maximum timestamp that is available among parents and $t_{max-last}$ is the t_{max} in the previous sliding window event. The regions are as follows:

- *Playing Region*: Blocks in this region are most likely received and any missing block should be requested and delivered during the current scheduling event[4].
- *Swarming Region*: Blocks in this region are partially delivered and a random subset of missing blocks in this sub-window should also be requested during this scheduling event.
- *New Region*: This region represents those blocks with the highest timestamps that have become available since the last scheduling event. These blocks are available only at the diffusion parent(s) and none of these blocks have been requested (and thus is not available) yet.

The block scheduling scheme at each peer is invoked once every Δ seconds and takes the following steps:

[4] Blocks from t_p till the start of playing sub-window are being played during this interval and should be already available in the buffer.

I) Quality Adaptation: it compares the average value of aggregate rate of data delivery ($\sum ewma\_dr(i)$) from all parents with the target quality (*i.e.*, the number of requested descriptions). If the aggregate rate of delivery is sufficient to accommodate another description, the target quality is increased by one description, *i.e.*, $IF\ C * (n+1) \leq \sum ewma\_dr(i)\ THEN\ n = n+1$. When the aggregate rate of delivery is not sufficient to sustain the current number of descriptions and the available buffer can not compensate this bandwidth deficit during one interval Δ, the target quality is reduced by one.

II) Requesting Diffusion Blocks: the scheduler requests any available blocks within the new region until all such blocks are requested or the bandwidth of the parent(s) are fully utilized. Note that only diffusion parents have blocks within new region. This strategy ensures rapid diffusion of new blocks to lower levels of the overlay.

III) Requesting Playing Blocks: Any missing blocks within the playing region is requested from the parents according to the scheduling and parent selection algorithm describing below.

IV) Requesting Swarming Blocks: the scheduler requests a subset of blocks in the swarming region that are available among parents and needed by the child. The requested blocks are determined in two steps as follows: *(i) Selecting Timestamps*: the scheduler determines the number of missing blocks for each timestamp within the swarming region by simply comparing the target quality with the number of unique blocks (from different descriptions) that it has already received for each timestamp.

This step generates a list of timestamps for blocks that can be pulled from swarming parents. *(ii) Assigning Blocks*: To select a random subset of required blocks, the scheduler shuffles the list of selected timestamps and sequentially examines each timestamp by taking two related actions:

- *Description Selection*: Determining a proper description such that the corresponding block (timestamp, description) is available among parents but missing at the child, and
- *Parent Selection*: Assigning the identified block to a parent that can provide it and has unused bandwidth.

The description for a given timestamp could be determined by selecting a *random* or *rarest* description from the useful descriptions among parents. The parent can be selected either randomly or based on the minimum ratio of its assigned blocks to its total block budget (*i.e.*, the fraction of its block budget that has been already assigned). Given the average bandwidth from each parent, we can estimate the total budget of each parent during one interval ($\frac{ewma\_bw(i)*\Delta}{PktSize}$). The latter parent selection criteria tends to proportionally balance the assigned blocks among parents during the scheduling process. The criteria and ordering for selection of description and parent of each required timestamp result in six variants of the scheduling scheme. We examine these six variants of scheduling in Section 3.6.3..

3.5. Source Behavior

Source plays a key role in controlling the diffused content to different diffusion subtrees (*i.e.*, level 1 peers). The maximum available quality in the system is determined by the average number of descriptions for each timestamp that are delivered from the source to all peers in level 1. The delivered quality to level 1 is determined by *(i)* the aggregate throughput from source to its child peers and *(ii)* the rate of delivery for new blocks from source to peers in level 1 which we call *diffusion rate*. For example, if the same block is requested (and thus sent) to multiple peers in level 1, the diffusion rate might be significantly lower than the aggregate bandwidth from source. In contrast, if all blocks are unique, the diffusion rate is equal to the aggregate bandwidth from source. The aggregate throughput from source is a function of its outgoing access link bandwidth coupled with its outgoing degree which is determined by $bandwidth-degree$ condition. Therefore, if source's access link bandwidth is equal to (or larger than) the stream bandwidth, it can deliver the full quality stream to the system if its aggregate bandwidth is properly used.

The number of unique descriptions from each timestamps and even distribution of diffused blocks in the overlay depend on requested blocks by peers in level 1. If the diffusion rate is equal to the stream bandwidth, then we observe proper behavior across lower levels since the blocks are simply multiplied by peer degree as they are pulled towards lower levels. However, in practice, due to independent requests by these peers from source, some blocks may never be requested while some other blocks may be requested multiple times. Moreover, the loss of delivered blocks to level 1 can decrease the diffusion rate.

Source is the only entity in the system, that can keep track of delivered blocks to each peer in level 1. Thus, source can minimize the redundancy in the requested blocks. This in turn *(i)* maximizes the delivered quality to level 1 (guarantee the diffusion of at least one copy of all blocks through the overlay) and *(ii)* evenly distributes delivered blocks across different timestamps and descriptions. To achieve this goal, in PRIME, source implements two related mechanisms as follows: First, it performs *loss detection* by keeping track of the number of successfully delivered blocks to peers in level 1. Second, source implements *block swapping* by keeping track of the number of delivered copies for each block and swapping any requested block with timestamp *ts* that has already been delivered with a block with the minimum number of delivered copies within timestamp window of $[ts{-}\Delta, ts]$. This strategy increases diffused quality from source through the overlay. We examine the source behavior with or without block swapping and loss detection in Subsection 3.6.2..

3.6. Performance Evaluation: Design Parameters

We use *ns* simulations to evaluate the effect of key design parameters on the performance of PRIME over a wide range of scenarios. Using packet level simulations has two important advantages compare to evaluation through experiments over a testbed such as PlanetLab as follows: *(i)* it enables us to investigate the effects of block level dynamics (and block loss) on system performance while capturing important details (*e.g.*, location of losses at different parts of an overlay). *(ii)* it allows us to construct a wide range of evaluation scenarios by controlling key variables such as peer properties (*e.g.*, level of bandwidth heterogeneity and asymmetry), resource availability and overlay connectivities.

A key challenge in the evaluation of PRIME is that changing a single parameter (*e.g.*, source bandwidth) may have multiple related (and potentially conflicting) effects on system performance. A unique feature of our evaluation is to carefully untangle multiple effects of important parameters.

Simulation Setting: We use the following default settings in our simulations: the physical topology is generated with Brite [126] using 15 ASs with 10 routers per AS in top-down mode, the overlay is directed, the bandwidth-degree condition is satisfied, and the delay on each access link is randomly selected between [5ms, 25ms]. Core links have high bandwidth (ranging from 4 to 10 Gbps) and thus all connections experience bottlenecks only on the access links. Furthermore, all connections are congestion controlled using RAP [124], and all routers use RED queue management.

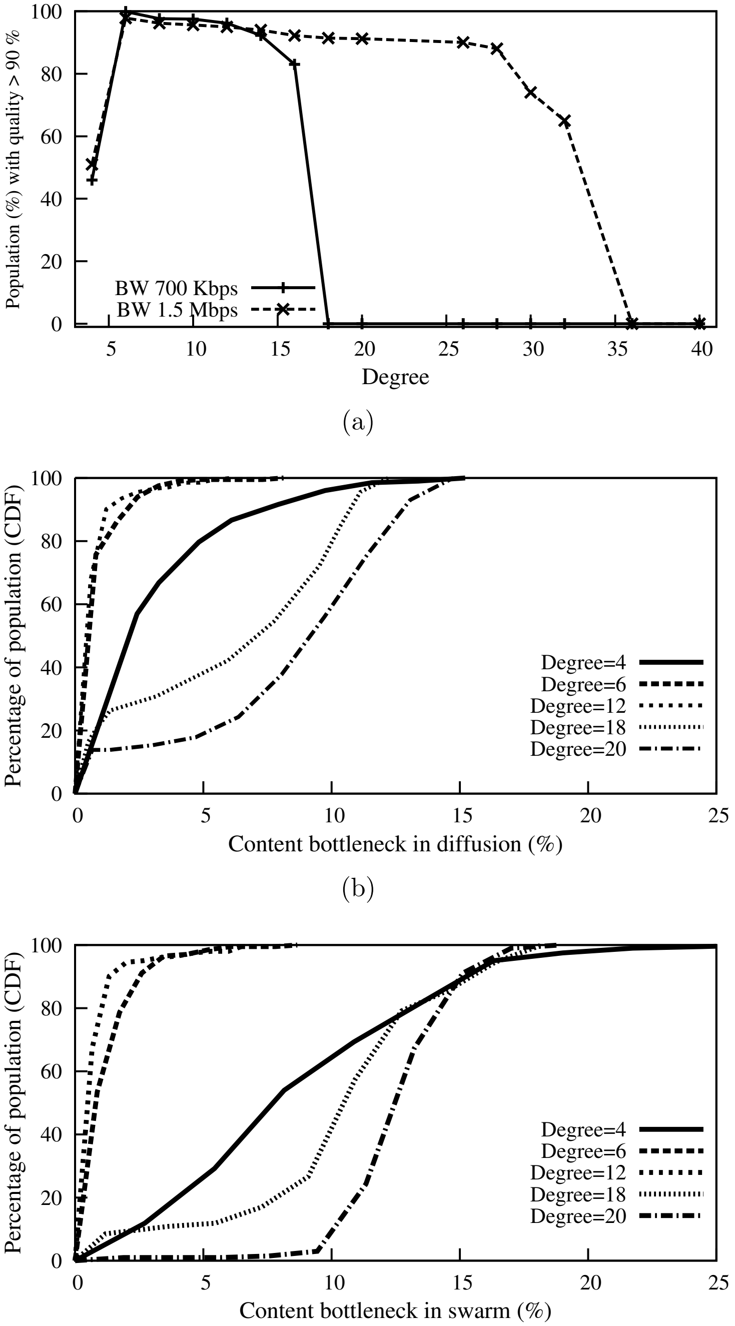
The delivered stream has 10 descriptions and all descriptions have the same constant bit rate of C = 160 Kbps. Source performs loss detection and block swapping. Each peer simulates the streaming consumption of delivered content after $\omega * \Delta$

seconds startup delay, and Δ is 6 seconds in all simulations[5]. Each simulation was run for 400 seconds. Our results represent the behavior of the system during the steady state after all peers have identified their parents and their pair-wise connections have reached their average bandwidth. Furthermore, our reported results are averaged across multiple runs of each scenario with different random seeds. We only focus on the *resource constraint* scenarios where supply is less than or equal to the demand for resources (*i.e.*, bandwidth), *i.e.*, resource index is less or equal to one. This allows us to stress test the protocol and ensures that the observed behavior is not a side effect of excess resources.

The following two scenarios are used as the *reference scenarios* in our evaluations: 200 homogeneous peers with *(i)* 700 Kbps and *(ii)* 1.5 Mbps access link bandwidth. Source bandwidth is set to the minimum value that ensures the delivery of sufficient stream quality ($\frac{peer_{bw}}{C}$) to the overlay. In the first scenario source bandwidth is 800 Kbps and in the second it is 1.6 Mbps.

We also use the following methodology to decouple and separately quantify the impacts of bandwidth and content bottlenecks on delivered content from each parent. Each parent always sends block to its children at the rate that is determined by a congestion controlled mechanism regardless of its useful content. At each block transmission time to a particular child, if there is an outstanding list of requested

[5] We note that Δ does not have a significant impact on system performance as long as it is sufficiently larger than RTT. We have examined different values for Δ and selected 6 seconds as a representative value.



(a)

(b)

(c)

FIGURE 7.: Effect of peer degree: (a) Percentage of peers that receive at least 90 percentile of the maximum quality across different degrees. (b) and (c) Distribution of content bottleneck across different degrees in diffusion and swarming phases, respectively.

blocks from that child, the outgoing block carries the first requested block in the list. Otherwise, the parent sends an especially marked block with the same size.

3.6.1. Peer Degree

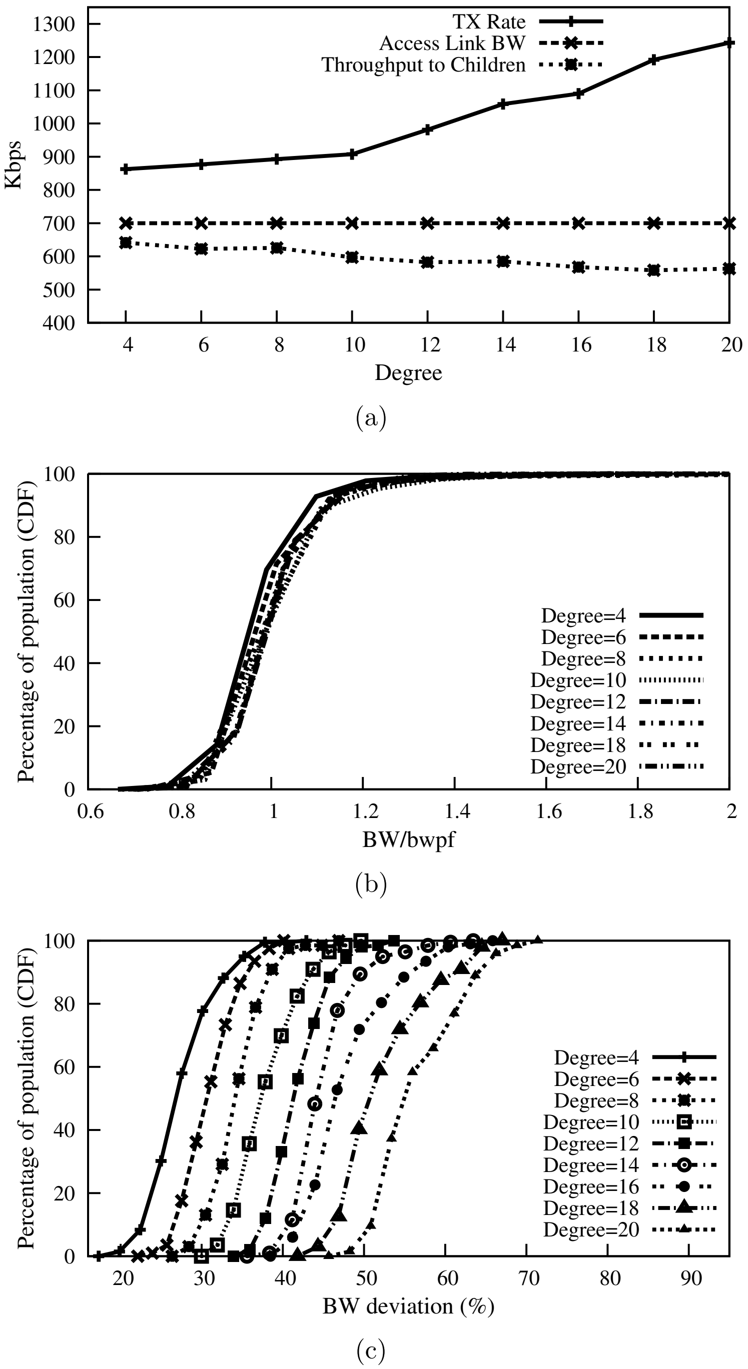
Our goal is to answer the following question: *"How does the connectivity of individual peers (*i.e.*, peer degree) affect the performance of content delivery in PRIME?"*. Given a group of peers with certain bandwidth, increasing peer degree improves the connectivity among peers but reduces the value of bandwidth-per-flow (or *bwpf*) for each connection. Figure 7.(a) depicts the percentage of peers that receive at least 90% of the maximum deliverable quality (*i.e.*, $\frac{inbw}{C}$) as a function of peer degree in the two reference scenarios. Note that for a fix population of peers, changing peer degree decreases the depth of the overlay. Therefore, for proper comparison, we adjust the value of ω based on the *depth* of each overlay as follows: $\omega = depth + 3$. The number of swarming intervals is constant across these simulations (K=3). Figure 7.(a) shows two interesting points: *(i)* in each reference scenario, there is a sweet range of peer degree over which a majority of peers receive a high quality stream, *(ii)* the sweet range of peer degree has the same lower bound (degree = 6) in both scenarios but its upper bound depends on the bandwidth-degree ratio.

The poor performance of the system for small peer degrees (degree<6) is due to the limited diversity of swarming parents which leads to content bottleneck among peers. When peer degree is small, the number of diffusion sub-trees will be proportionally small because of the bandwidth-degree condition. This in turn proportionally reduces the probability that the randomly selected swarming parents for each peer would be located on different diffusion sub-trees and thus increases the probability of content bottleneck among peers regardless of peer bandwidth. The rapid drop in the delivered quality for large peer degrees is the result of significant increase in loss rate of individual connections. Figure 7.(a) clearly shows that the upper bound for the reference scenario with peer bandwidth 1.5 Mbps is almost twice as the the upper bound for peer bandwidth 700 Kbps. This demonstrates that the upper bound of the sweet range of peer degree is indeed a function of loss rate rather than the peer degree. We examine the effect of loss rate for higher peer degrees in further details later in this section.

To verify our explanation, Figures 7.(b) and 7.(c) depict the distribution of content bottlenecks from the diffusion and swarming parents among peers with peer bandwidth 700 Kbps for a few peer degrees, respectively. The percentage of content bottleneck from the diffusion (or swarming) parents is the percentage of congestion controlled bandwidth from the diffusion (or swarming) parent(s) that is not utilized for content delivery (*i.e.*, the percentage of delivered blocks that are especially marked). Comparing Figures 7.(b) and 7.(c) shows that the percentage of content bottleneck is clearly higher from the swarming parents across all degrees which agrees with our discussion in subsection 3.4.3.. Furthermore, as we increase the peer degree from 4 to 6, the percentage of content bottleneck in both phases significantly decreases due to the improved diversity among swarming parents. However, any further increase in peer degree (beyond 12) reverses this trend and rapidly increases the percentage of content bottleneck in both phases due to the increase in loss rate.

Loss Rate: To further examine the effect of connection loss rate on system behavior for large peer degrees, Figure 8.(a) plots (from top to bottom) the aggregate transmission rate from a parent to all of its children, the parent's access link bandwidth and aggregate throughput to all of its children. The gap between the top two lines shows the bandwidth associated with lost blocks at the outgoing access link of the parent peer whereas the gap between the bottom two lines represents the bandwidth associated with lost blocks at the incoming access link of all children, collectively. This figure shows that the aggregate throughput from a parent peer to all of its children drops with increasing peer degree. More interestingly, while losses mostly occur at the parent's outgoing access link, a non-negligible fraction of losses also occur at the incoming access link of children as well. This suggests that throughput of some connections are limited by the parent's outgoing access link bandwidth while others are limited by the child's incoming access link bandwidth.

We further investigate the effect of loss rate by examining the distribution of normalized average throughput (normalized by the corresponding *bwpf*) and its deviation across all connections for different peer degrees in Figure 8.(b) and 8.(c), respectively. These two figures paint an insightful picture on how the dynamics of congestion controlled bandwidth affect the location of bottleneck for individual



(a)

(b)

(c)

FIGURE 8.: Effect of peer degree: (a) Transmission rate of a selected peer along with its access link bandwidth and aggregate throughput to all of its children. (b) Distribution of BW/*bwpf* values across all connections. (c) Distribution of the deviation of aggregate bandwidth across all peers.

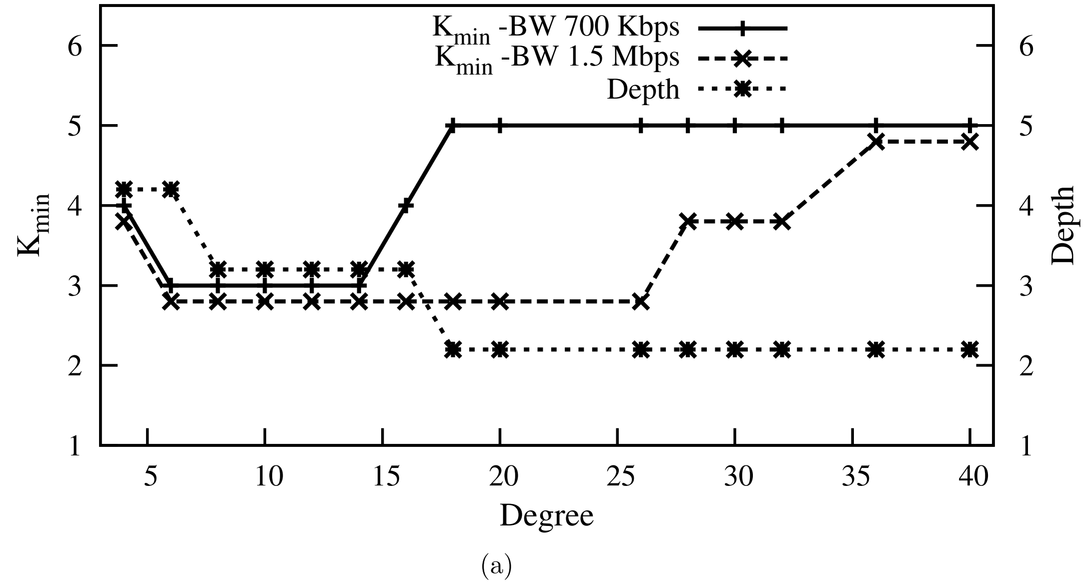
connections. As peer degree increases, the distribution of normalized average throughput across all connections does not change but the distribution of its deviation shifts towards higher values. The larger deviation in per-connection bandwidth with larger peer degrees result in bottlenecks at both sender and receiver ends of individual connections. This in turn reduces the throughput of individual connections which causes bandwidth bottleneck for the corresponding child peers, and content bottleneck for all the descendant peers[6].

Buffer Requirement: The poor performance outside the sweet range of peer degree indicates that the number of swarming intervals is inadequate for the delivery of the required number of data units to most of the peers due to the content bottleneck. This raises the following question: *"How many swarming intervals are required in a given scenario so that the majority of peers receive a high quality stream?"*. Figure 9.(a) depicts the number of diffusion intervals (*i.e.*, *depth*) and the minimum number of required swarming intervals ($K_{min} = \omega_{min}$-*depth*) as a function of peer degree in both reference scenarios (labeled as K_{min}-) such that 90% of peers receive 90% of the maximum deliverable quality. Figure 9.(a) shows that the depth of the overlay is independent of the peer bandwidth and gradually decreases with peer degree. As peer degree increases, K_{min} initially decreases from 4 to its minimum value of 3 intervals

[6] Conducting similar simulations with TFRC revealed that TFRC exhibits a lower loss rate but results in even lower utilization than RAP. In summary, when peer degree is large, an aggressive congestion control mechanism such as RAP may cause a rather higher loss rate and thus content bottleneck whereas slower congestion control mechanisms such as TFRC reduce the loss rate at the cost of lower utilization of resources.

within the sweet range of peer degree due to the increasing diversity in the location of swarming parents across different diffusion subtrees. However, further increase of peer degree beyond a threshold results in the increase in K_{min} due to the higher loss rate and the resulting increase in content bottleneck which requires a longer swarming phase. In essence, this figure demonstrates *(i)* the minimum buffer requirement for individual peers (in a given scenario) in terms of the number of intervals as a function of peer degree (*i.e.*, ω_{min}=*depth*+K_{min}), and *(ii)* the direct relationship between K_{min} and *bwpf* for different peer degrees.

Pattern of Content Delivery: We investigate the effect of peer degree on the pattern of content delivery by examining the following question *"How does the distribution of the average path length (in hops) among delivered blocks to individual peers change as peer degree increases (*i.e.*, the overlay becomes more connected)?"* . Figure 9.(b) presents this distribution for average path length among peers for several peer degrees in the reference scenario with peer bandwidth 700 Kbps when the number of swarming intervals is equal to K_{min}. This figure reveals the following two important changes in the average path length to individual peers as overlay connectivity improves: *(i)* the average path length to individual peers monotonically decreases with peer degree primarily due to the decrease in overlay depth, *(ii)* the distribution of average path length among peers becomes more homogeneous. This is due to the increase in the diversity of swarming parents which in turn evens out the probability of content bottleneck among peers. The increasing homogeneity of average path length with



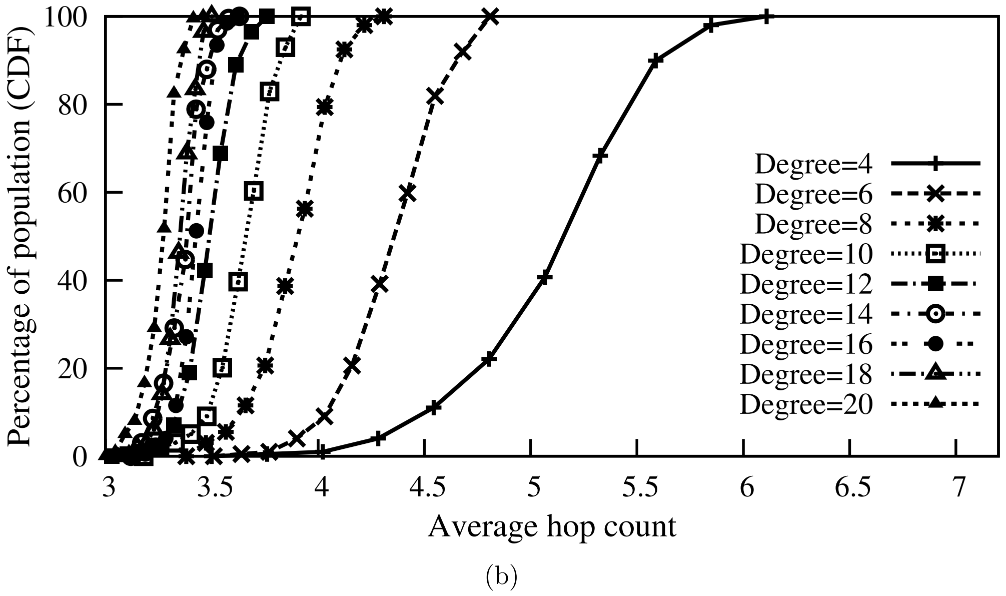
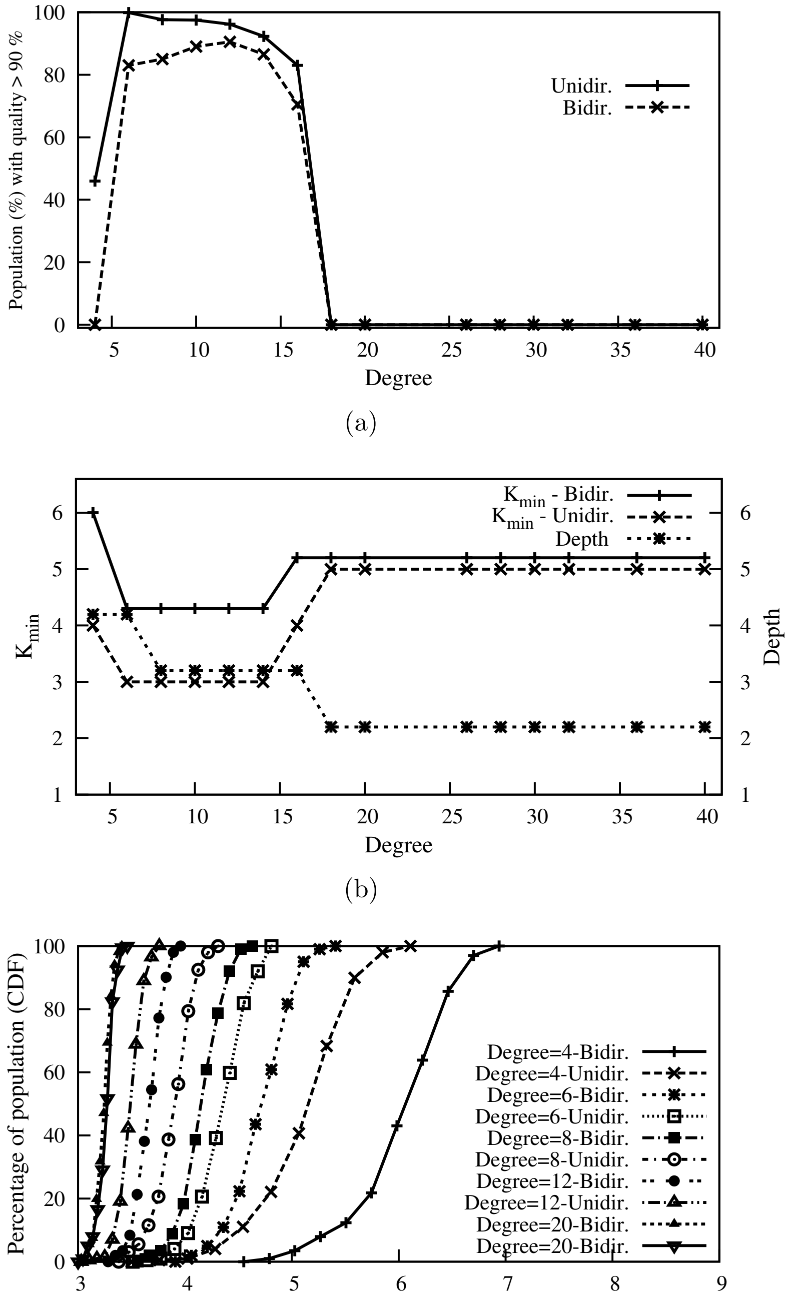


FIGURE 9.: Effect of peer degree: (a) K_{min}, and *depth* across different peer degrees with peer bandwidth of 700 Kbps and 1.5 Mbps. (b) Distribution of average path length across different peer degrees with K_{min}.

peer degree also implies that lost blocks are requested from the same parent during the following swarming interval(s) rather than through a longer path from other swarming parents.

Bi- vs Uni-directional Connectivity: Maintaining bi-directional connections between peers affects their connectivity. This raises the following question *"Is the performance of content delivery different over an undirected overlay (and if so, why)?"*. To investigate this issue, we examine the reference scenario with 700 Kbps bandwidth but enforce bi-directional connections among peers. Figure 10.(a) shows the percentage of peers that receive 90% of the maximum deliverable quality over such a bidirectional overlay as a function of peer degree when K_{min} is 3 (labeled as Bidir.). This figure reveals that the percentage of peers with high quality in a bi-directional overlay is 10%-20% less than the uni-directional overlay over the sweet range of peer degree. Figure 10.(b) also shows the value of K_{min} for these bidirectional overlays as a function of peer degree. Figure 10.(b) indicates that bi-directional overlays require at least one extra swarming interval for peer degrees between 4 and 16. To explain this result, we note that bi-directional connections reduce the number of swarming shortcuts among diffusion sub-trees and thus increase the percentage of content bottleneck during the swarming phase. More specifically, for each diffusion connection from a parent to a child, there is a swarming connection in the reverse direction that connects two peers within the same diffusion sub-tree which is not an *effective* swarming shortcut. In a bidirectional overlay, effective swarming shortcuts between different sub-trees are



(a)

(b)

(c)

FIGURE 10.: Effect of bi-directional overlays: (a) Percentage of peers that receive at least 90 percentile of the maximum quality across different degrees. (b) K_{min}, and *depth* across different peer degrees. (c) Distribution of average path length across different degrees with K_{min}.

established through connections between peers in the same level. Since most such "intra-level" connections are located at the bottom level, peers in higher levels of the overlay require a larger number of swarming intervals. Figure 10.(c) depicts the distribution of average path length for the above bidirectional overlays as well as the corresponding unidirectional overlays (that were shown in Figure 9.(b)) for easy comparison. This figure indicates that the distribution of average path length over the bi-directional overlay is around one hop longer than the uni-directional overlay for peer degree of 4. However, the difference in path lengths between bi- and uni-directional overlays rapidly diminishes with increasing peer degree. Note that the number of ineffective swarming shortcuts is roughly equal to the number of diffusion connections which is a function of the number of peers. Therefore, for a fixed population, as the peer degree increases, the extra connections must establish useful swarming shortcuts. This in turn improves the diversity of swarming parents and reduces the average hop count (and its deviations) for individual peers as shown in Figure 10.(c).

3.6.2. Source Behavior

In this subsection, we examine the effect of the following two orthogonal aspects of source behavior on the system performance: *(i)* Block swapping and loss detection, and *(ii)* Source bandwidth.

Block Swapping & Loss Detection: We explore the effect of source coordination in the reference scenario with 700 Kbps access link bandwidth where source bandwidth and K_{min} are 800 Kbps and 3, respectively. This configuration ensures all peers in level 1 receive a high quality stream. Figure 11.(a) depicts the delivered quality from source to level 1 (*i.e.*, diffusion rate to level 1) as a function of peer degree in three different scenarios: *(i)* source without any coordination, *(ii)* source with only block swapping, and *(iii)* source with both block swapping and loss detection. Note that the outgoing bandwidth from source is fully utilized across these scenarios and its aggregate throughput to level 1 is not affected by the coordination mechanism. Figure 11.(a) shows that the diffusion rate slowly decreases with peer degree in all three scenarios due to the increase in loss rate (as we described in Figure 7.(a)). Incorporating block swapping significantly increases the diffusion rate, and adding loss detection leads to further improvement in the diffusion rate. Figure 11.(b) depicts the distribution of the number of delivered copies for individual blocks to level 1 in the above three scenarios when peer degree is 10. This figure clearly illustrates that incorporating block swapping and then loss detection progressively balances out the number of copies of delivered blocks to level 1. *In summary, incorporating block swapping and loss detection enable us to deliver certain quality with a lower source bandwidth or to improve delivered quality for a given source bandwidth.*

Notation	Method	Parent selection	Description selection
$ParentMin. - Desc.Rare$	Parent first	Minimum $\frac{assigned}{totalbudget}$	Rarest
$ParentMin. - Desc.Rand$	Parent first	Minimum $\frac{assigned}{totalbudget}$	Random
$ParentRand - Desc.Rand$	Parent first	Random	Random
$ParentRand - Desc.Rare$	Parent first	Random	Rarest
$Desc.Rare - ParentMin.$	Description first	Minimum $\frac{assigned}{totalbudget}$	Rarest
$Desc.Rand - ParentMin.$	Description first	Minimum $\frac{assigned}{totalbudget}$	Random

TABLE 5.: Summary of different block scheduling schemes.

3.6.3. Block Scheduling

In Section 3.4.4., we presented the criteria for description selection (*i.e.*, random, rarest) and parent selection (*i.e.*, least proportionally loaded, random) and the relative order of selection (between description and parent) as basic design choices for block scheduling scheme. These choices lead to six variants of the block scheduling scheme. In this subsection, we compare the performance of these six variants of the scheduling in the reference scenario with access link bandwidth of 700 Kbps and assume that all peers use the same block scheduling scheme. We examine the following scheduling schemes:

(i) $ParentMin. - Desc.Rare$,

(ii) $ParentMin. - Desc.Rand$,

(iii) $ParentRand - Desc.Rand$,

(iv) $ParentRand - Desc.Rare$,

(v) $Desc.Rare - ParentMin.$ and

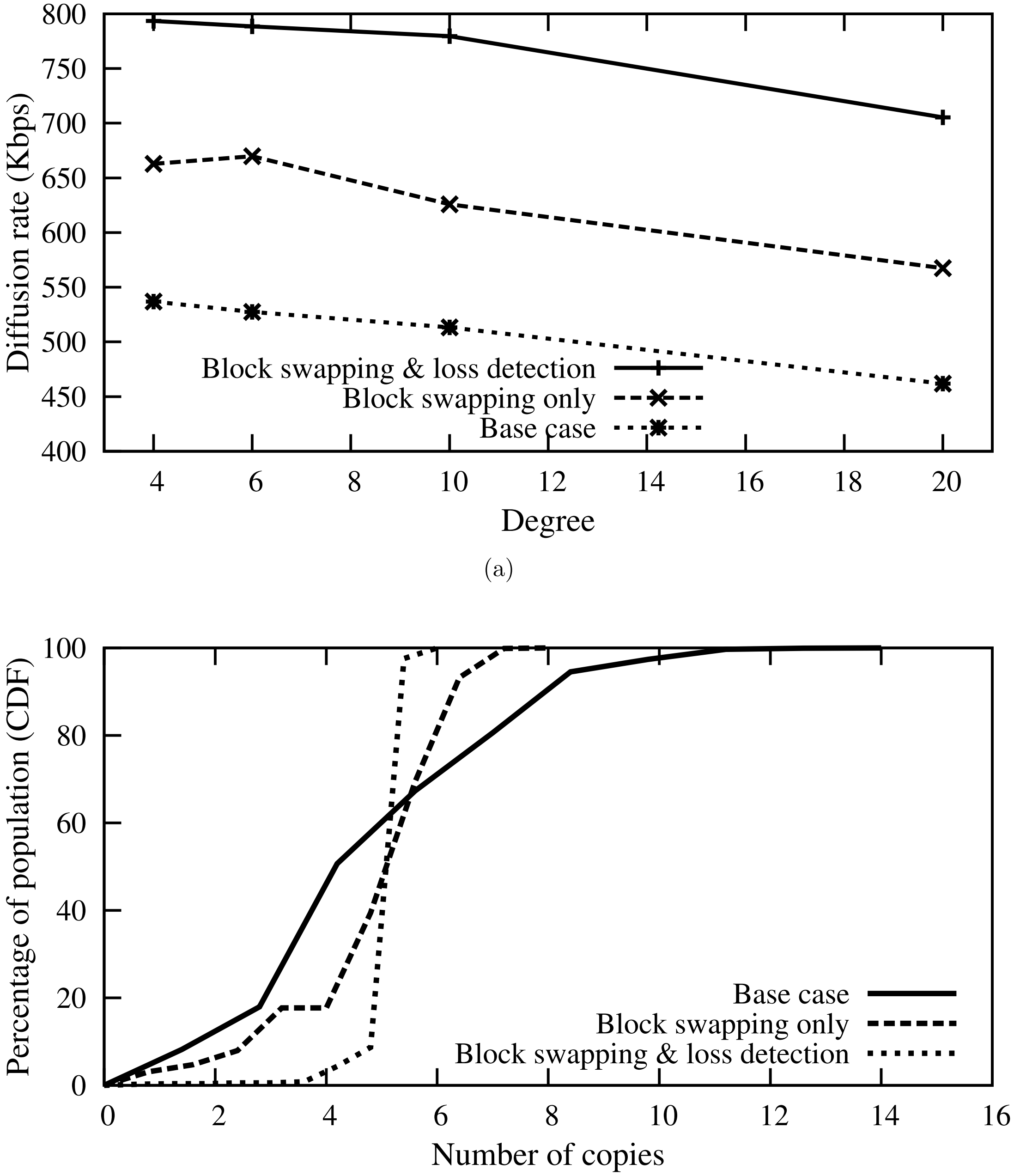
(vi) $Desc.Rand - ParentMin.$.

Table 5. summarizes the notations based on the method that is used and the corresponding criteria for different block scheduling schemes.

Figure 7.(a) depicts the percentage of peers that receive 90% of the maximum deliverable quality as a function of peer degree for these six block scheduling schemes where $\omega = depth + 3$. This figure illustrates two interesting points: First, except for the two scheduling schemes that randomly select the parent, the performance of other schemes is very similar within the sweet range of peer degree. *This suggests that neither the criteria for selecting the description of a block nor the relative order of selection (between description and parent) significantly affects the performance of block scheduling schemes.* Second, the percentage of peers that receive a high quality stream in the two low-performing schemes (labeled as $ParentRand - Desc. Rand/Rare$) is very similar, and roughly 20% lower than other schemes within the sweet range of peer degree. Selection of parents regardless of the selection criteria for description performs roughly 20% lower than other mechanisms. Figure 12.(b) shows the minimum swarming intervals (K_{min}) where 90% of peers receive 90% of their maximum deliverable quality. This figure revealed that the K_{min} value for $ParentRand$ mechanisms is always one interval larger than other scheduling schemes in a comparable scenario. Figure 12.(c) presents CDF of percentage of content bottleneck from swarming parents for different block scheduling schemes for degree of 12. This figure shows that the percentage of content bottleneck from swarming parents for $ParentRand$ selection mechanisms is significantly larger than other mechanisms.

Intuitively, those schedulings which request a block from a random parent are more likely to experience content bottleneck due to the higher frequency of *deadlock*s during parent selection. A deadlock event occurs when a required block is available among some parents but it can not be requested since the bandwidth budget of those parents are fully allocated for delivery of other blocks. To verify this hypothesis, Figure 13.(a) depicts the distribution of frequency of deadlock (*i.e.*, the fraction of blocks that experience deadlock during the scheduling process) among peers for all six schedulings when peer degree is 12. Figure 13.(a) clearly shows that the median frequency of deadlock is roughly four times higher for schedulings that use random parent selection. The random parent selection may not request all the unique blocks from individual parents. Therefore, a fraction of bandwidth budget from diffusion parents is used for the delivery of blocks that are already available at other parents.

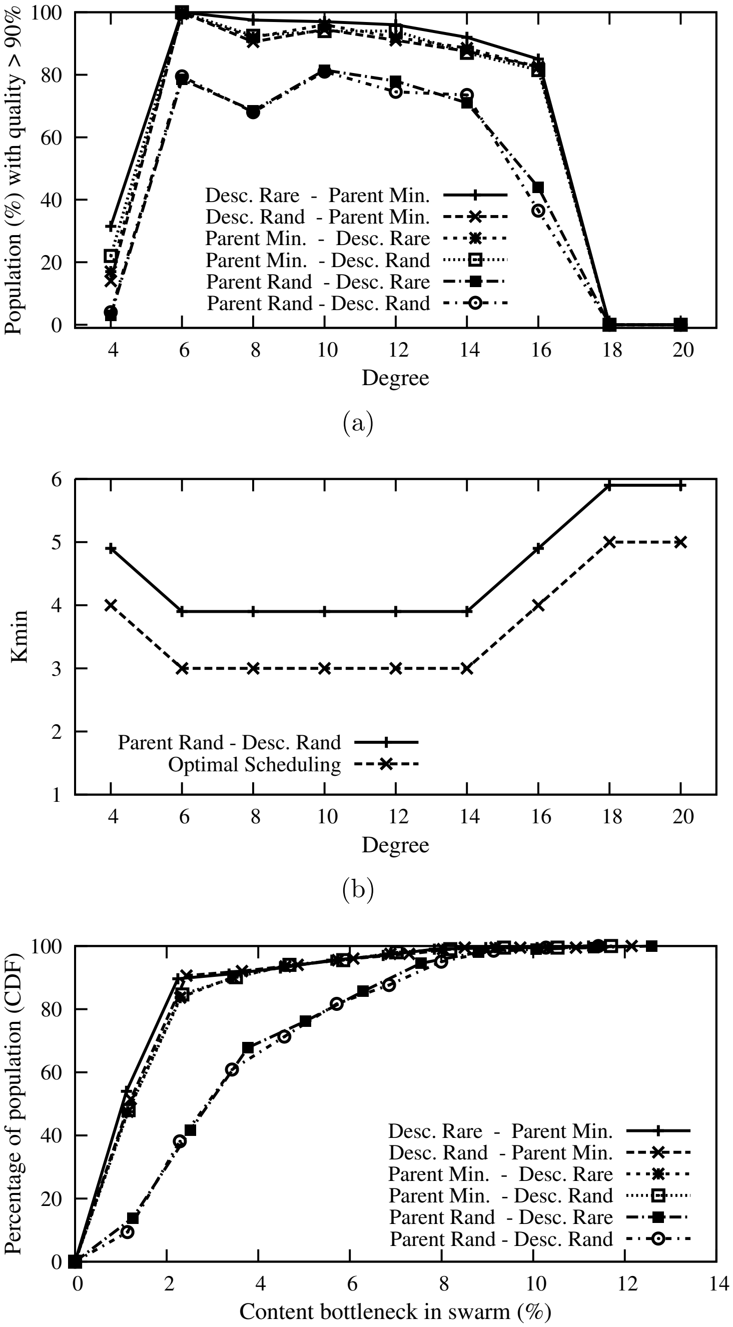
Closer examination of the two low-performing scheduling schemes reveals that these two schemes can achieve good performance with an extra swarming interval (*i.e.*, larger buffer, ω). This raises the following interesting question *"Does extra swarming intervals accommodate the delivery of deadlocked blocks through longer paths to reduce the frequency of deadlock?"*. Figure 13.(b) depicts the distribution of average path length (in hops) across delivered blocks for one of the high-performing scheduling scheme ($ParentMin. - Desc.Rand$) as a reference and one of the low-performing scheduling scheme ($ParentRand - Desc.Rand$) (with a proper number of swarming intervals) across different peer degrees. Figure 13.(b) reveals that the average path



(a)

(b)

FIGURE 11.: Effect of source behavior: (a) Diffusion rate to the level 1 across various peer degrees in three scenarios with various source behavior.. (b) Distribution of the number of copies of each block that are delivered to level 1 for peer degree 10 when source behavior changes.



(a)

(b)

(c)

FIGURE 12.: Effect of scheduling schemes: (a) Percentage of peers with at least 90 percentile delivered quality across different peer degrees. (b) K_{min} for a good performing and "Parent Random, Description Random" scheduling across various peer degrees. (c) Distribution of content bottleneck in swarming phase.

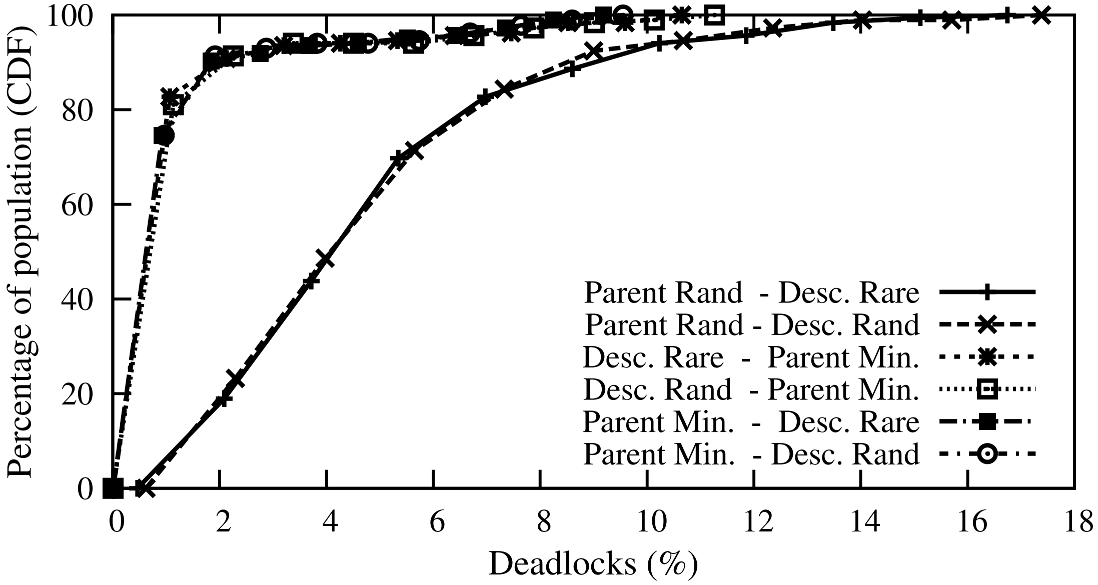
length for the low-performing scheduling scheme with longer swarming is around 20% longer for all peer degrees. This suggests that 20% of peers that have poor performance in Figure 12.(a), can leverage the extra swarming interval to request the deadlocked blocks from another swarming parent. The larger number of swarming intervals increases the pool of swarming blocks and decreases the probability of deadlock event.

3.7. Performance Evaluation: Overlay Properties

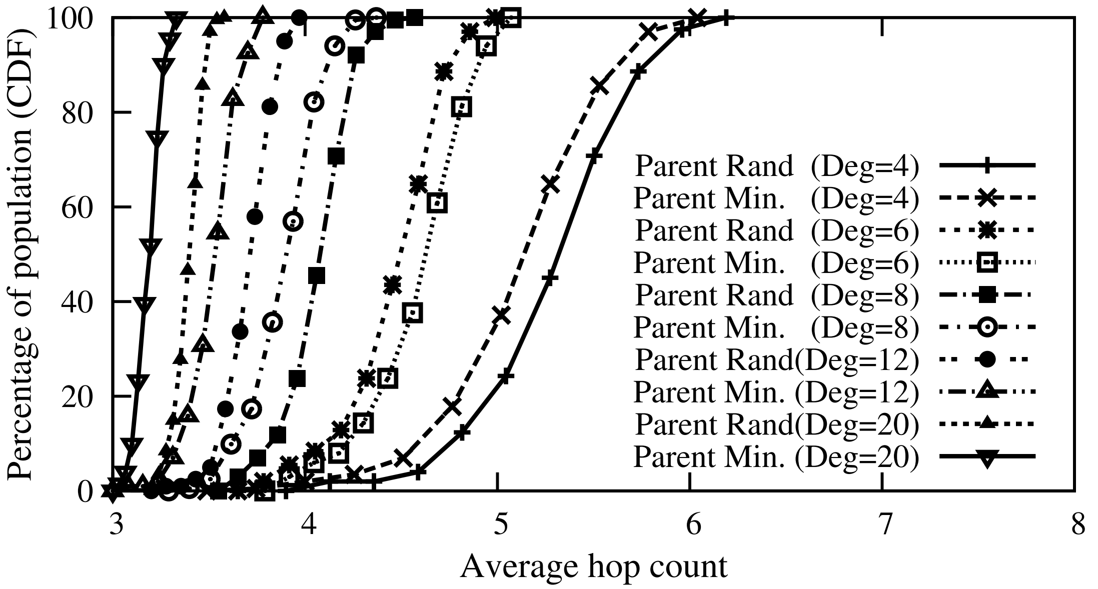
In this section, we examine the effect of overlay properties on the performance of PRIME. We focus on the peer bandwidth heterogeneity, peer population, churn and the effect of resources.

3.7.1. Peer Bandwidth Heterogeneity

To investigate the effect of bandwidth heterogeneity, we consider the reference scenario with peer bandwidth 1.5 Mbps (bw_h) and reduce the access link bandwidth for a fraction of peers to bw_l. As we showed in subsection 3.3.1., the bandwidth-degree condition ensures a high utilization of access link among all peers even when peers have heterogeneous bandwidth. The percentage of content bottleneck for low bandwidth peers in heterogeneous scenarios is lower than homogeneous scenarios since some of their swarming parents are likely to be high bandwidth peers with higher available



(a)



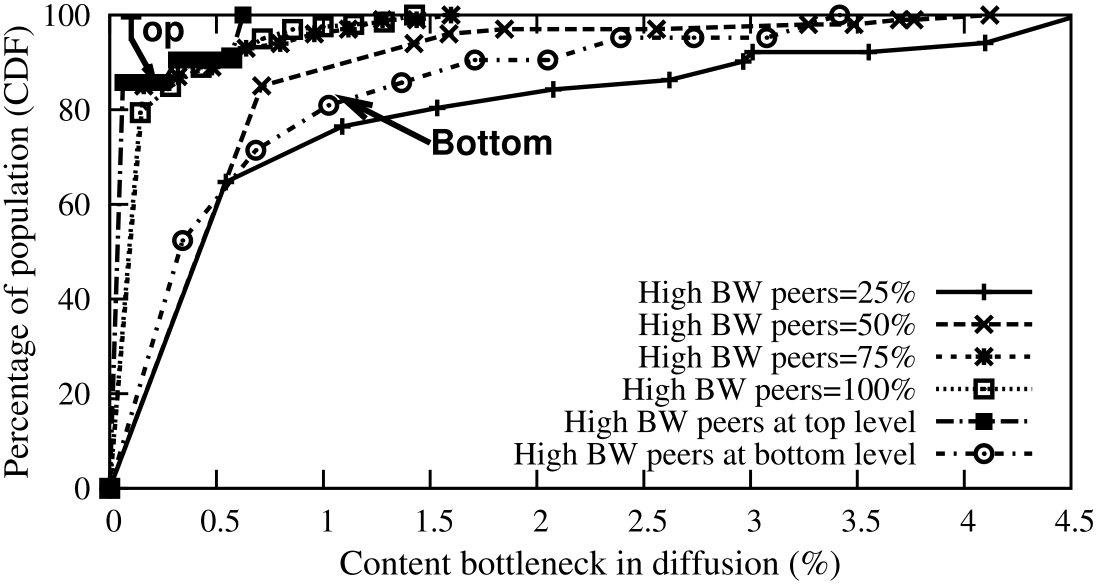
(b)

FIGURE 13.: Effect of scheduling schemes: (a) Distribution of the frequency of deadlocks for peer degree of 12 across various scheduling schemes. (b) Distribution of average path length for two high and low performing scheduling schemes across different degrees.

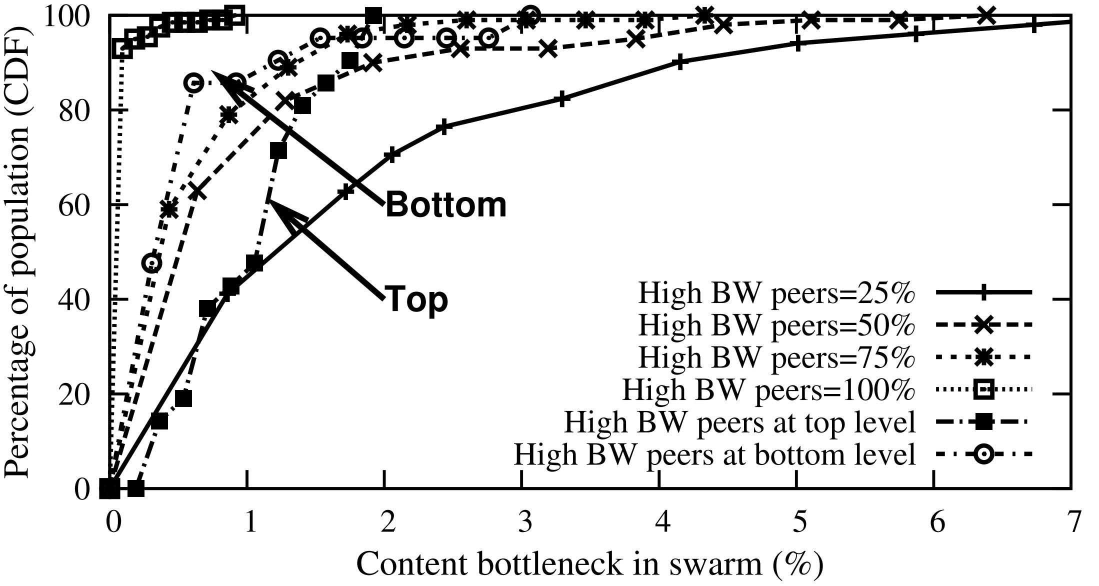
quality. Therefore, we focus on the delivered quality to high bandwidth peers. The first question is: *"How are the delivered quality and buffer requirement of high bandwidth peers affected by the percentage of low bandwidth peers?"*.

Figures 14.(a) and 14.(b) show the distribution of content bottleneck among high bandwidth peers (bw=1.5 Mbps) with different percentage of low bandwidth peers (1 Mbps) from diffusion and swarming parents, respectively. These figures show that the percentage of high bandwidth peers has a minor impact on the content bottleneck in both phases. Figure 14.(a) and 14.(b) show a minor increase in content bottleneck from the diffusion and swarming parents when the percentage of high bandwidth peers is small. In the diffusion phase, this is due to the decrease in the total number of overlay connections and the resulting increase in the overlay depth. In the swarming phase, the percentage of content bottleneck at each peer depends on the aggregate available content among its swarming parents. As the number of high bandwidth peers decreases, a larger fraction of their swarming parents are likely to be low bandwidth peers. This in turn reduces the aggregate available quality among their swarming parents and increases the probability of content bottleneck among high bandwidth peers. We have also examined other scenarios with different levels of bandwidth heterogeneity ($\frac{bw_h}{bw_l}$) and observed that the level of heterogeneity does not have any impact on the delivered quality to high bandwidth peers.

Location of High Bandwidth Peers: Another important question in an overlay with heterogeneous peers is: *"How does the location of high bandwidth peers in the*



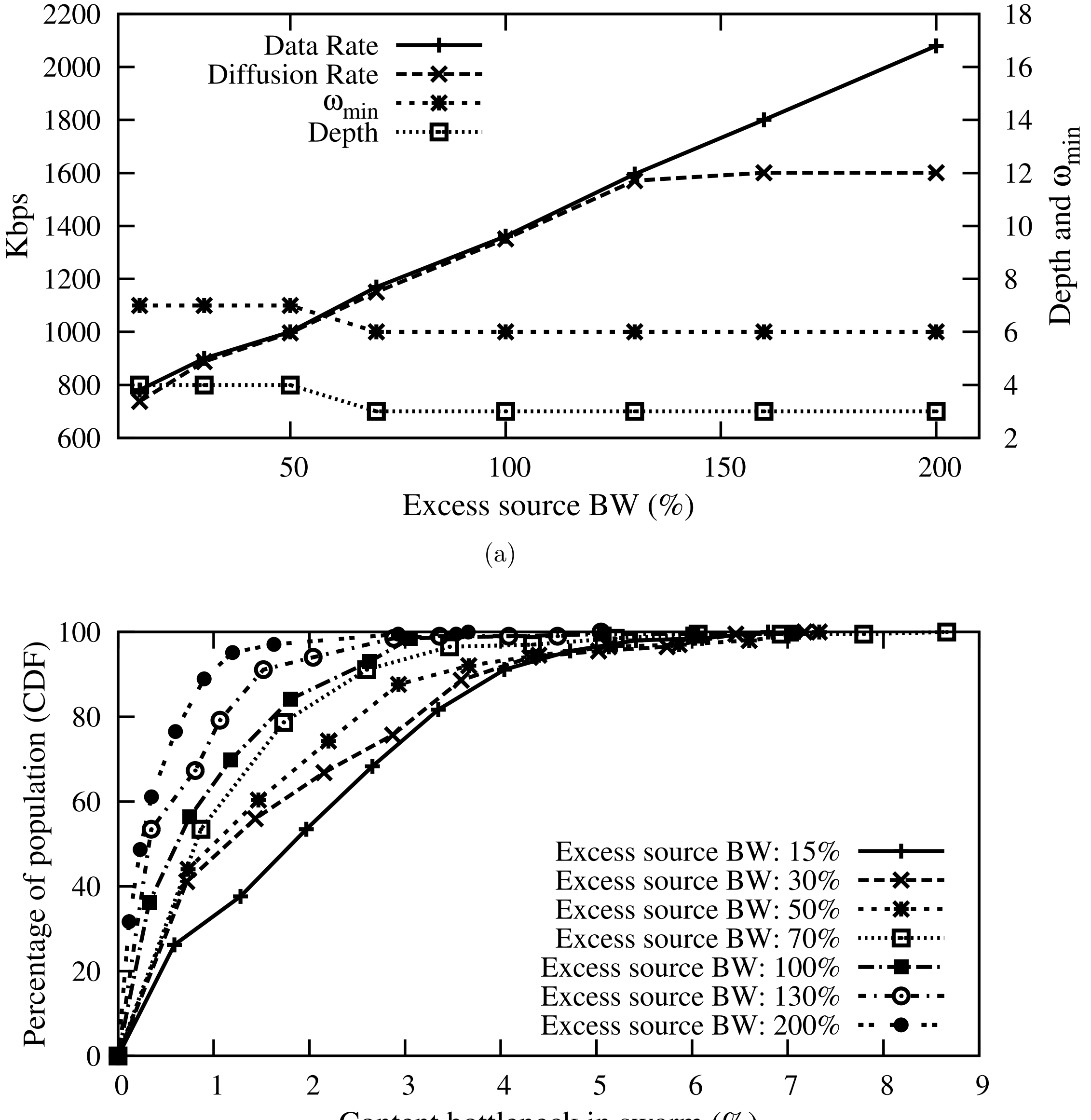
(a) Diffusion



(b) Swarm

FIGURE 14.: Effect of bandwidth heterogeneity: (a) and (b) Distribution of content bottleneck among high bandwidth peers in heterogeneous scenarios from diffusion and swarm parents, respectively.

overlay affect the percentage of content bottleneck among them?". To examine this issue, we explore a heterogeneous scenario where only 10% of peers have access link bandwidth of 1.5 Mbps and the remaining peers have access link bandwidth of 1 Mbps. We enforce the overlay construction mechanism to only place high bandwidth peers at the top level (as source's children) or at the bottom level. Figures 14.(a) and 14.(b) show the percentage of content bottleneck for these two cases (labeled as "top" and "bottom") for comparison with previous scenarios. Placing the high bandwidth peers in non-bottom levels reduces the depth of the overlay and thus reduces the required number of diffusion intervals. However, it also reduces the connectivity among the diffusion sub-trees and thus increases the probability of content bottleneck from the swarming parents. In contrast, placing high bandwidth peers at the bottom level slightly increases overlay depth and thus increases the content bottleneck in diffusion phase. However, this effect is compensated by the higher connectivity among the diffusion sub-trees which decreases the probability of content bottleneck from the swarming parents. *In summary, the location of high bandwidth peers in the overlay has an opposite effect on the probability of content bottleneck in diffusion and swarming phases. Therefore, the overall impact on the performance of content delivery and the minimum buffer requirement (*i.e., *ω) is relatively small.*



(a)

(b)

FIGURE 15.: Effect of source bandwidth: (a) Diffusion rate and data rate to level 1 peers along with ω and depth across various excess source bandwidth. (b) Distribution of content bottleneck across various excess source bandwidth from the swarming parents.

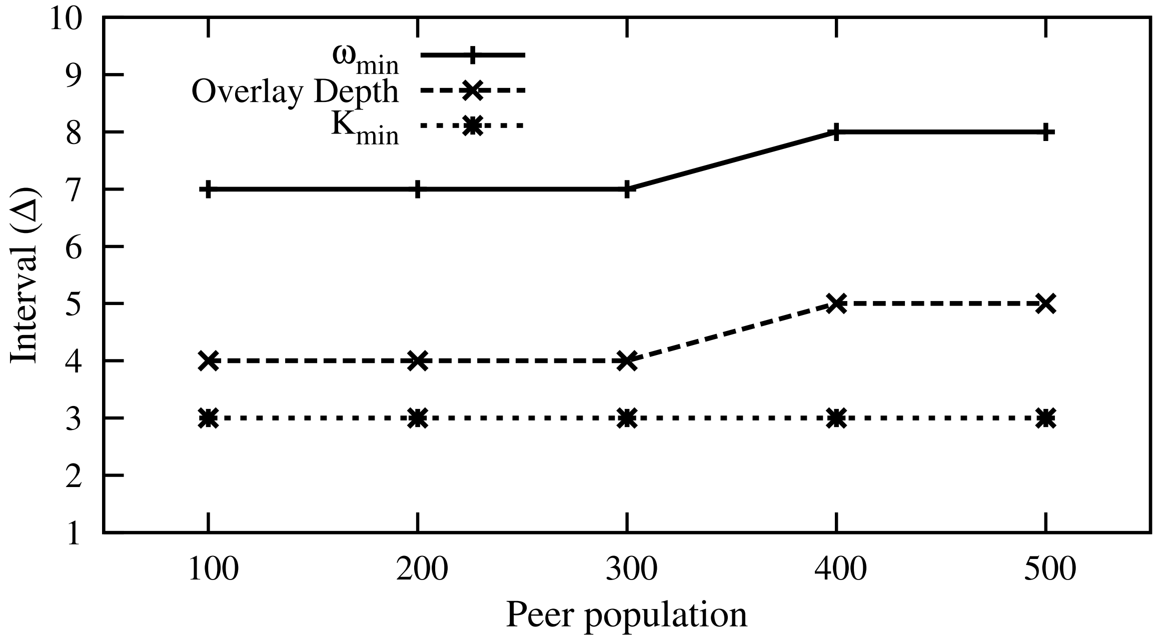


FIGURE 16.: Effect of peer population: K_{min}, *depth* and ω_{min} for different peer populations.

adding redundant diffusion sub-trees (that do not have unique content). This reduces overlay depth and slightly reduces content bottleneck during the diffusion phase.

3.7.3. Peer Population

We examine the scalability of PRIME protocol by addressing the following question: *'How do the delivered quality and buffer requirement at individual peers change with peer population?"*.

Figure 16. shows the duration of diffusion phase (or overlay *depth*), the minimum duration of swarming phase (K_{min}) and the minimum buffer requirement (or ω_{min})

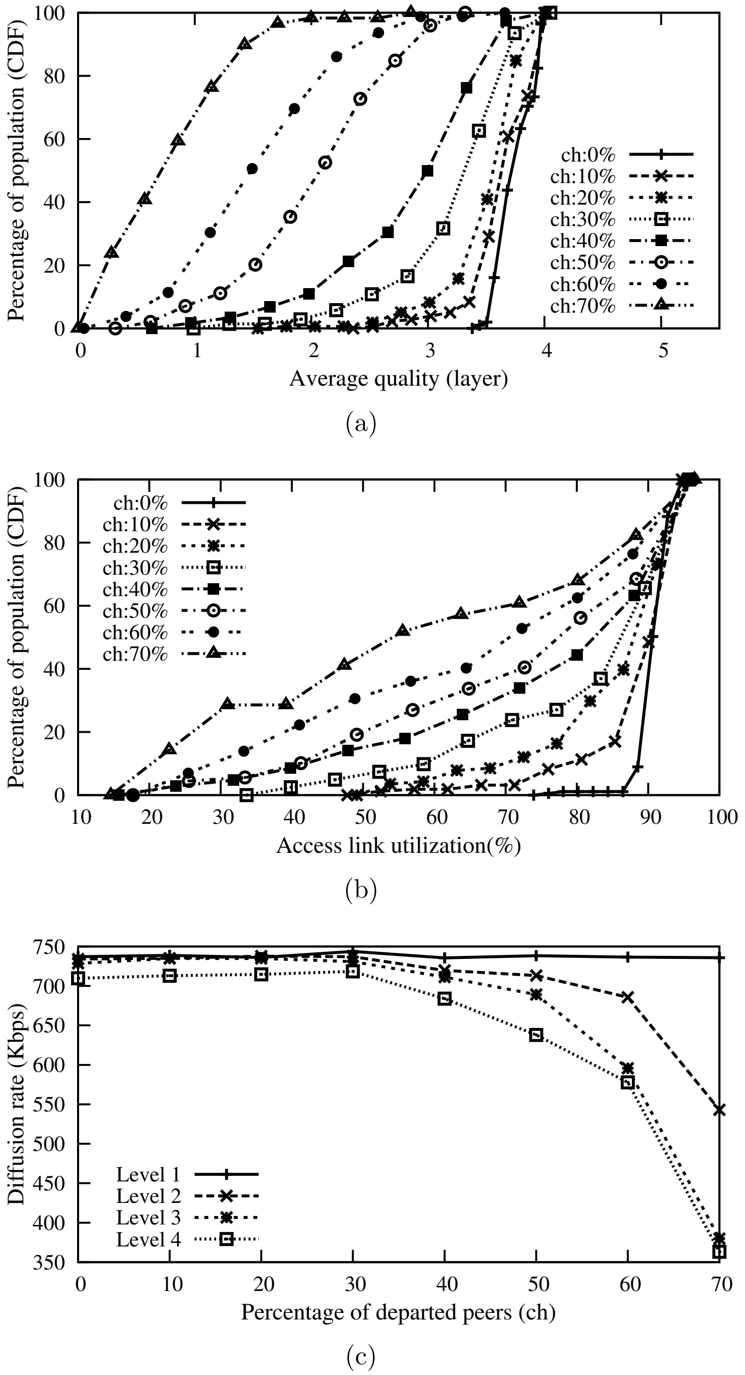
as a function of peer population in the reference scenario with access link bandwidth of 700 Kbps when peer degree is 6. This figure provides a good evidence of the scalability of PRIME with user population. As the peer population increases, overlay depth slowly grows but the duration of the swarming phase (with a proper peer degree) remains constant. To explain this, we note that increasing peer population does not affect the number of diffusion sub-trees. This means that the diversity of swarming parents for individual peers does not change with peer population. Therefore, the observed content bottleneck and the required number of swarming intervals for individual peers does not change with peer population. We have observed the same behavior for different degrees within the sweet range of peer degree. *The observed trend in this result suggests that within the sweet range of peer degree, PRIME can effectively utilize available resources in the system and provide maximum quality to peers in a scalable fashion if the buffer size is logarithmically increased with peer population.*

3.7.4. Peer Dynamics

So far we have not considered the effect of peer dynamics (or churn) in our simulations. In practice, churn may have both short-term (or transient) and long-term effects on the performance of content delivery in PRIME. When a peer leaves the overlay, the aggregate bandwidth to its children is dropped until each child manages to establish a connection to a new parent. The transient effect of parent departure on delivered quality to a child depends on the efficiency of the parent discovery (*i.e.*, time to connect to a new parent) and the amount of buffered content at the child among other things. Over a longer term, churn could change the bandwidth-to-degree ratio among peers in the overlay. We call such an overlay a *distorted* overlay where the bandwidth-to-degree condition is not satisfied. Initially, we focus on this long-term effect of churn on the performance of content delivery since it is more significant than the transient effect and it does not depend on protocol-specific details. Further, we present our result on the short-term impact of churn on the performance of PRIME.

3.7.4.1. Long-term Impact of Churn on Delivery

To examine the performance of content delivery over a distorted overlay, we consider the reference scenario with peer access link bandwidth of 700 Kbps, peer degree 6 and $\omega = depth + 3$ where bandwidth-degree condition is satisfied. We emulate a distorted overlay by removing $ch\%$ of randomly selected peers from the reference scenario without allowing remaining peers to establish new connections. We can control the level of distortion by changing the percentage of departed peers (ch). The resulting distorted overlay represents the snapshot of the overlay structure as peers join and leave the system. As the level of distortion increases, the distribution of peer population across different levels of the overlay becomes more imbalanced compare to a properly connected overlay and the depth of the overlay may increase.



(a)

(b)

(c)

FIGURE 17.: Effect of long-term peer dynamics: (a) Distribution of average delivered quality (layers) to each peer for various percentages of *ch*. (b) Distribution of utilization of access link bandwidth among peers for various percentages of *ch*. (c) Diffusion rate to different levels for various percentages of *ch*.

Figure 17.(a) depicts the distribution of average delivered quality among peers for different levels of distortion. This figure reveals that the delivered quality to peers is rather sensitive to the level of distortion and rapidly drops as *ch* passes 30%. One key question is *"Is the decrease in delivered quality due to the drop in the utilization of access link bandwidth (*i.e.*, bandwidth bottleneck) or the inability of peers to utilize the available bandwidth (*i.e.*, content bottleneck)?"*. Figures 17.(b) shows the distribution of incoming access link utilization among peers for different levels of distortion. This figure indicates that the utilization of access link bandwidth drops with the number of departed peers. However, comparing Figures 17.(a) and 17.(b) illustrates that the decrease in delivered quality is visibly larger than the drop in access link utilization when level of distortion in the overlay is roughly larger than 30%. This suggests that both bandwidth and content bottleneck contribute into the drop in quality as the overlay becomes more distorted.

To identify the underlying causes for content bottleneck in distorted overlays, we examine average diffusion rate at each level of the overlay as distortion increases in Figure 17.(c). Figure 17.(c) demonstrates that the diffusion rate at the top level is not affected by the percentage of departed peers as long as the number of peers in level 1 is not affected. However, the diffusion rate at all lower levels is rapidly dropped once more than 30% of peers depart. A closer examination of the overlay connectivity revealed that when a large fraction of peers depart, some diffusion sub-trees may become disconnected (especially at the higher levels) from the rest of the overlay , *e.g.*, a peer in level 1 does not have any child. Such an event has a ripple effect and reduces the diffusion rate to all the lower levels of the overlay due to the content bottleneck. This implies that increasing the number of swarming intervals does not improve delivered quality in these scenarios. We have conducted simulations with longer buffer sizes and confirmed this observation. *In summary, as the overlay becomes more distorted, the delivered quality to individual peers is dropped due to both bandwidth and content bottleneck. The content bottleneck is caused by the disconnection of some diffusion sub-trees from the rest of the overlay.*

3.7.4.2. Short-term Impact of Churn on Delivery

In this subsection we examine the short term effect of peer dynamics on PRIME. We choose a good degree of 6 and we set ω equals to 7 ($\omega = depth + 3$). At time 250 sec we remove 10% of peers from the overlay without reconstructing the overlay.

As mentioned before, peer dynamics in PRIME might have short effects on delivered quality to individual peers. Short term effects of losing a parent on a peer are *(i)* missing blocks that are requested from that particular departed parent during last window and *(ii)* draining buffer until the peer adapts its quality *e.g.*, number of layers it wants to play.

Figure 18. shows the evolution of the estimated weighted moving average aggregate bandwidth and data rate for a random peer which lost its parent at time

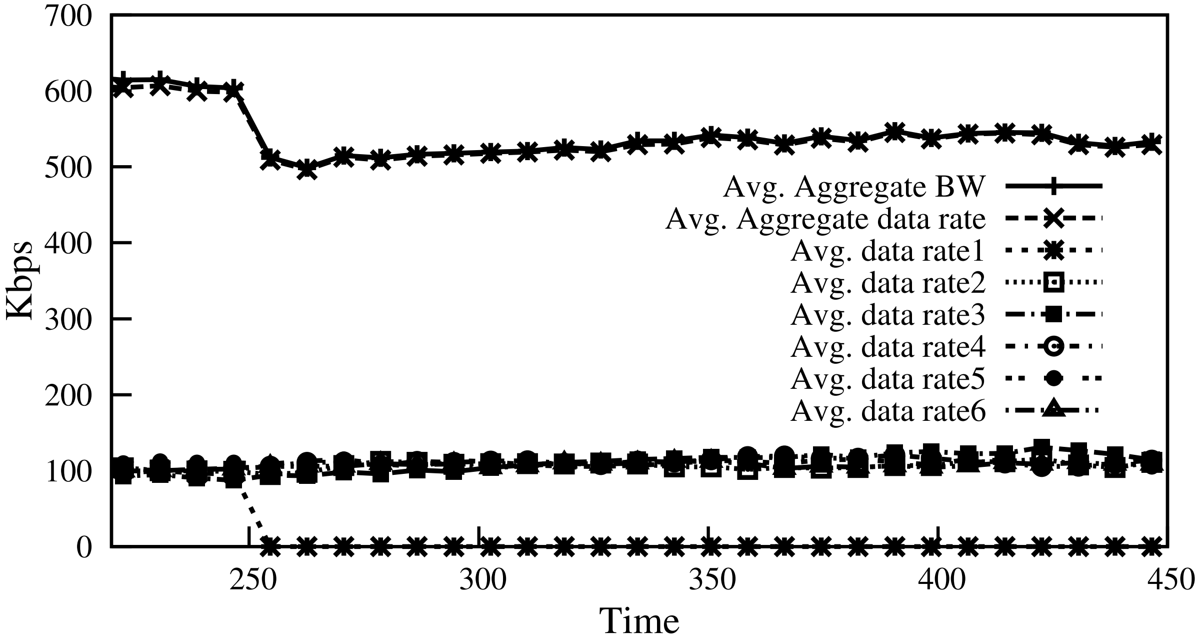


FIGURE 18.: Effect of short-term peer dynamics: Evolution of estimated weighted moving average bandwidth and data rate over time. Aggregate bandwidth and data rate are shown along with data rate of each parent. At time 250 sec, first parent departed.

250 sec. We can clearly observe that the aggregate bandwidth and data rate drop around time 250 sec. This figure also depicts that the aggregate data rate closely follows the aggregate bw over time which confirms that despite loss of a parent content bottleneck does not increase and individual peers still can achieve very high utilization of bandwidth.

Uplink BW	SC1	SC2	SC3	SC4	SC5	SC6
$128kbps$	27%	54%	13%	5%	11%	50%
$384kbps$	60%	20%	80%	9%	14%	39%
$1000kbps$	13%	26%	7%	36%	25%	11%
$0kbps$	0%	0%	0%	50%	50%	0%
RI	1	1	1	1	0.8	0.8

TABLE 6.: Target scenarios with peers of different outgoing access link bandwidth.

3.7.5. Resources, Bandwidth Asymmetry & Free-riders

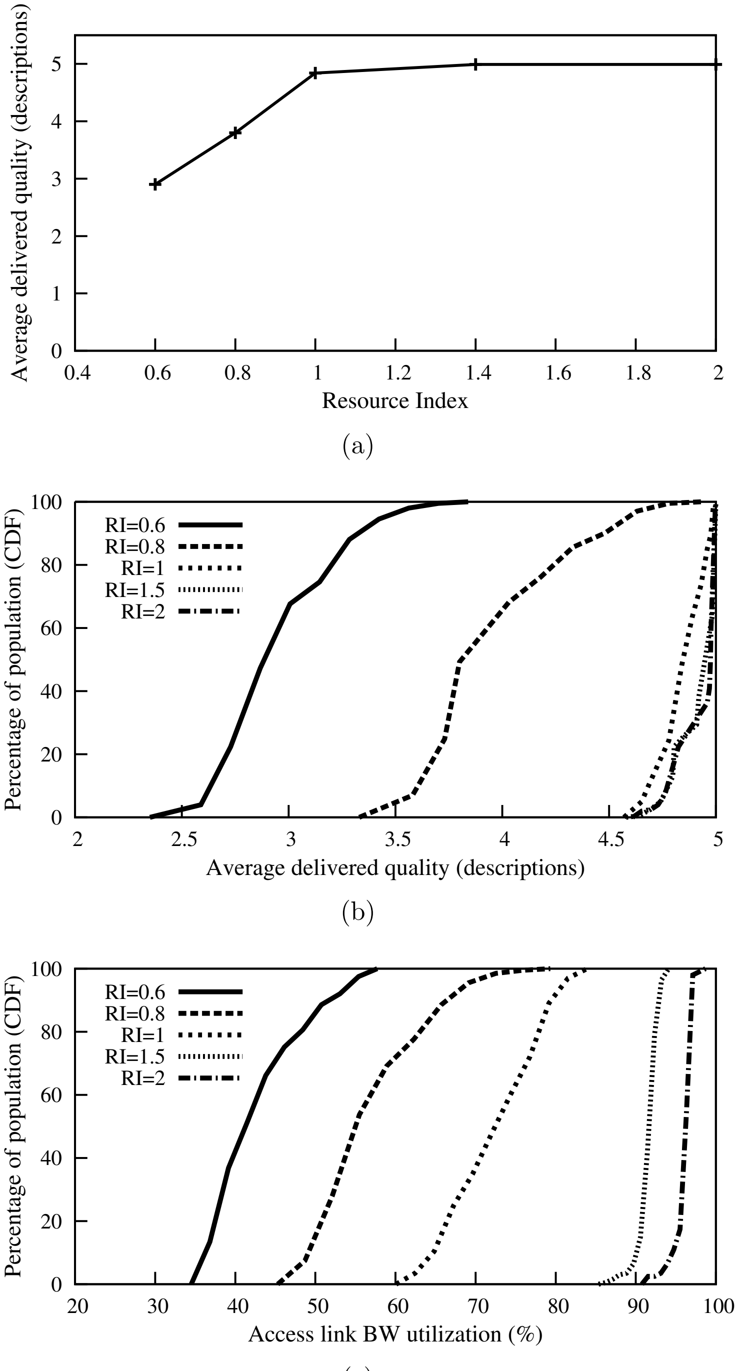
In all the previous subsections, we assumed that participating peers have symmetric access link bandwidth and their downlink bandwidth is equal to the stream bandwidth. In such a scenario, the aggregate demand and supply for bandwidth are equal, and the ratio of demand to the supply for bandwidth which is called *resource index* (RI), is one. In practice, the uplink bandwidth that a peer is able or willing to contribute might be less than its incoming bandwidth. Therefore, the aggregate resources may not be sufficient to provide maximum deliverable quality to all peers. In such a resource-constraint scenario, the key question is *"Is the drop in quality fairly similar across participating peers?"*.

To examine the effect of aggregate available resources, bandwidth asymmetry and free-riders we control the outgoing access link bandwidth of participating peers as follows: The outgoing bandwidth of individual peers can be set to one of the four values: 128 Kbps, 384 Kbps, 1000 Kbps and 0 Kbps. The rate of full quality version of the stream is 400 Kbps. The incoming access link bandwidth of all peers are set to 550 Kbps so that each peer can easily receive the full quality playback rate. The incoming access link bandwidth of all peers are set to 700 Kbps to receive the fully quality stream. By controlling the distribution of peers across these four groups, we can control the heterogeneity of outgoing access link bandwidth, percentage of free riders (with outgoing bandwidth of zero) which in turn determines the aggregate outgoing bandwidth (*i.e.*, system capacity) for a given scenario. Each column in Table 6. shows the distribution of 200 peers across different groups which represents one of our target scenarios. Table 6. presents the value of RI for each target scenario.

Effect of Available Resources- To examine the effect of available resource, we consider a scenario where all participating peers have the same outgoing bandwidth and then change the value of outgoing access link bandwidth to vary the value of RI from 0.6 to 2.

Figure 19.(a) depicts the *averaged* delivered quality (in terms of number of descriptions) to participating peers as a function of RI. This figure clearly shows that when RI<1, the average delivered quality is less than maximum quality but it increases with RI. Peers receives the full quality in average as long as RI is larger then one. When the resource index is less than one, the received quality is proportional to the resource index.

Figure 19.(b) provides a more detailed view by showing the distribution of delivered quality to individual peers for different RI values. When RI is greater than one, a majority of peers receive full quality stream. But when RI drops below one, the distribution of delivered quality becomes more skewed among participating peers. For



(a)

(b)

(c)

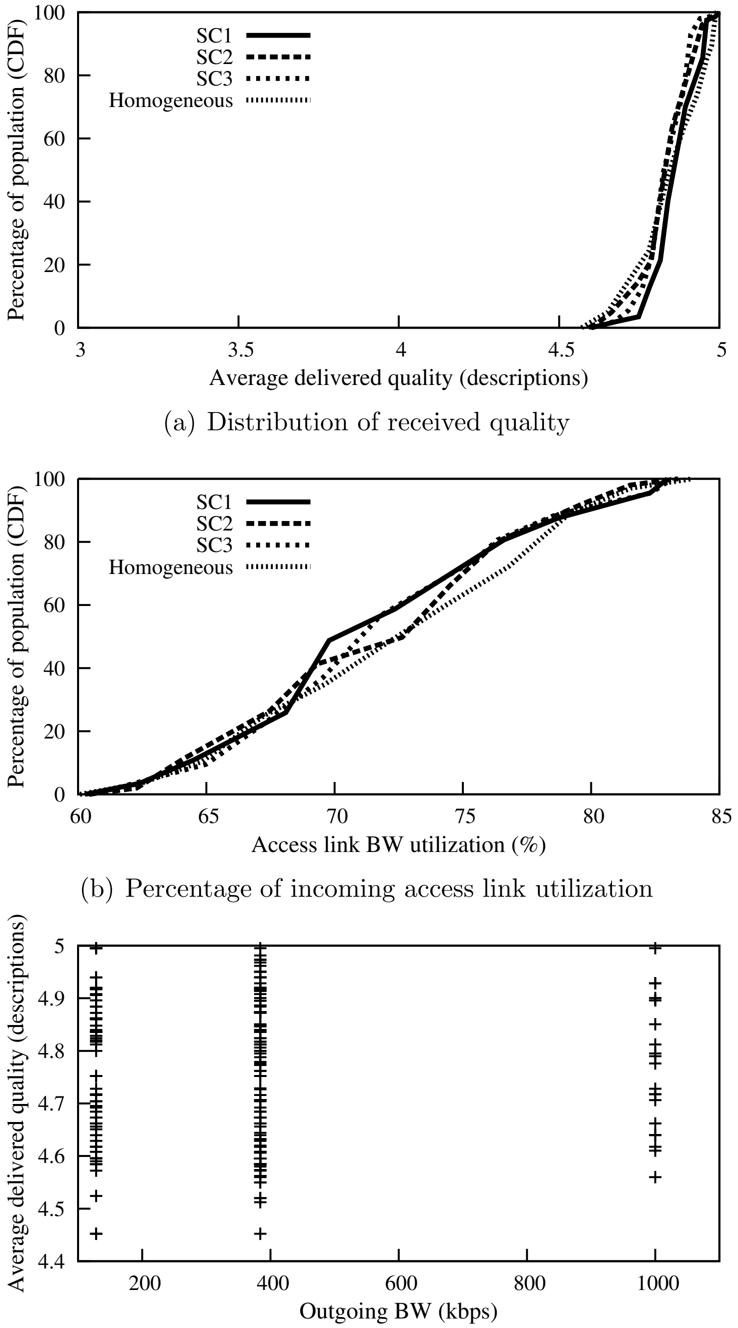
FIGURE 19.: Effect of resources: (a) Average delivered quality to individual peers for various RI levels. (b) Distribution of delivered quality to each peer for various RI levels. (c) Distribution of access link bandwidth utilization of peers for various RI levels.

instance, when RI is 0.6, the range of delivered quality is between 2.3 to 4 descriptions while all peers have the same incoming bandwidth.

To identify the underlying causes for the skewness of delivered quality when RI is less than 1, we examine the distribution of access link bandwidth utilization among peers as shown in Figure 19.(c). This figure shows that as RI drops below 1, the utilization of access link bandwidth becomes skewed similar to delivered quality. This in turn indicates that the bandwidth bottleneck is the main factor in the diversity of delivered quality to different peers. More specifically, when RI<1, some peers luckily obtain a larger portion of available resources because their aggregate bandwidth from their parents is higher due to the dynamics of congestion control mechanism.

Effect of Heterogeneous Uplink Bandwidth- We now examine the impact of heterogeneous uplink bandwidth on the performance of P2P streaming. Towards this end, we focus on scenarios SC1, SC2 and SC3 in Table 6.. While these three scenarios have the same RI value of 1, their distribution of uplink bandwidth is different. Figure 20.(a) and 20.(b) depict the CDF of delivered quality and utilization of access link bandwidth among participating peers for these three scenarios. These figures show that the degree of heterogeneity in uplink bandwidth does not affect the distribution of delivered quality to participating peers.

We also examine the correlation between the delivered quality and the contributed resources (*i.e.*, outgoing bandwidth) of participating peers. Figure 20.(c) depicts a scattered plot of the delivered quality to individual peers as a function of their



(a) Distribution of received quality

(b) Percentage of incoming access link utilization

(c) Correlation between contribution and received quality

FIGURE 20.: Effect of bandwidth asymmetry and resources: (a) and (b) Distribution of delivered quality and access link bandwidth utilization for scenarios 1,2 and 3 and homogeneous case as a comparison, respectively. (c) Average delivered quality versus outgoing bandwidth of peers in SC1.

uplink bandwidth. This figures illustrates that there is no correlation between the contributed resource by each peer and the quality it receives (*i.e.*, the amount of resources it consumes). This is clearly an improper behavior since it does not provides an incentive for higher bandwidth peers to participate. To address this issue, P2P streaming mechanism should incorporate a contribution-aware resource allocation mechanism such that the allocated resource to individual peer would be proportional with their contributed resources.

Effect of Free riders- A key challenge in any P2P system is to gracefully accommodate (or at least limit the potential damage by) uncooperative peers that do not contribute any resource (*i.e.*, free-riders). We investigate the impact of free-riders by examining scenarios SC4 and SC5 with RI values of 0.8 and 1, respectively, while half of participating peers are free-riders as shown in Table 6..

Figure 21. shows the distribution of delivered quality to participating peers in scenarios SC4 and SC5. We have also included the distribution of delivered quality for peers with homogeneous outgoing bandwidth when RI is 0.8 and 1 as references for comparison. Figure 21. reveals that the presence of free riders significantly reduces the delivered quality even when the aggregate available resources (*i.e.*, the value of RI) remains intact. We note that the scenario with free riders can be viewed as a special case for bandwidth heterogeneity. Therefore, the significant drop in delivered quality as the result of free rider was rather surprising since the heterogeneity of bandwidth does not have a major effect on performance as we reported previously.

A closer examinations of our results revealed that the free riders affect the connectivity of the overlay in such a way that they disrupt the two phase content delivery. This behavior can be explained as follows: Since free riders have no child peers, their presence in the overlay can affect the connectivity (and thus exchange of content) between different diffusion subtrees. For example, if peer 10, 11 and 12 in Figure 5. are free riders their corresponding diffusion subtree will not be connected to the swarming mesh (does not have any swarming connection). More specifically, a diffusion subtree that only has free riders at its bottom level cannot provide its corresponding content to peers on other diffusion subtrees. This in turn limits the delivered quality to other participating peers. However, peers on such a disconnected diffusion subtree can still receive the content from other diffusion subtree due to the directed nature of connectivity in the overlay. Even if the diffusion subtrees do not get completely disconnected from the swarming mesh, the presence of free riders could increase the depth of the overlay which in turn affect the buffer requirement at each peer. *In summary, our results reveal that the presence of free riders can significantly affect the connectivity between different diffusion subtrees in the overlay which in turn prevents content swarming among them and thus limits the delivered quality to a subset of peers.* This suggests that P2P streaming mechanism should ensure proper connectivity among participating peers (*e.g.*, monitoring the overlay) to address such significant drop in quality. Clearly, identifying free riders and removing them from

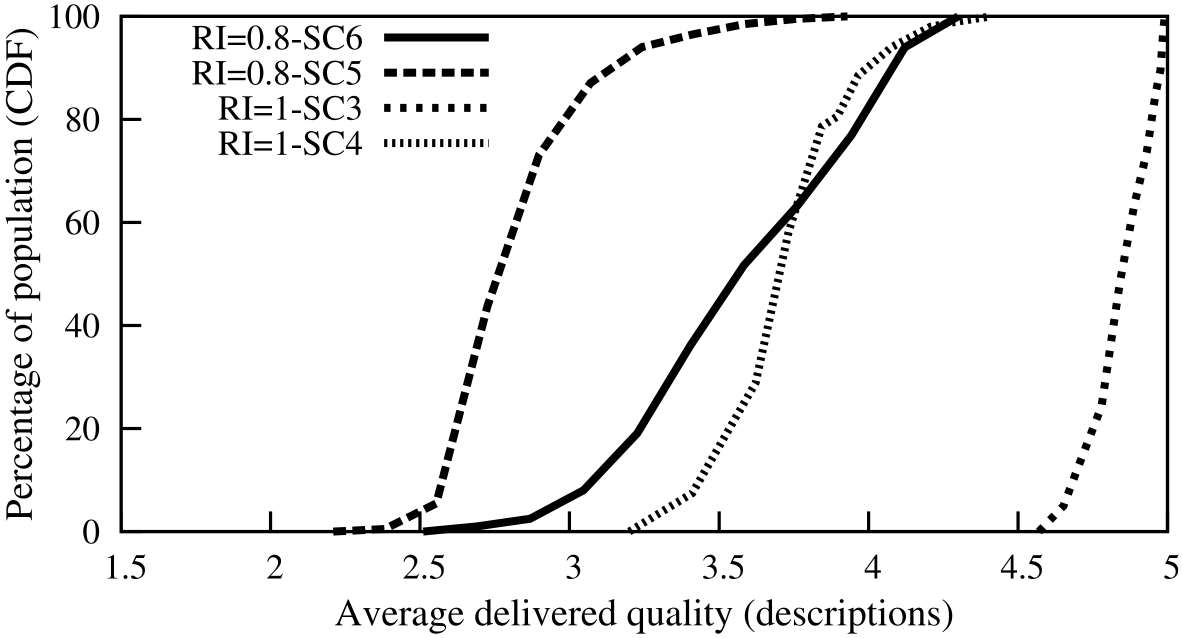


FIGURE 21.: Effect of free riders on delivered quality.

the overlay would also help but it is non-trivial in such a distributed environment without central authority.

3.8. Summary

This chapter presents PRIME, a novel mesh-based P2P streaming mechanism for live content that can effectively incorporate swarming content delivery. We designed PRIME through a performance-driven approach by identifying underlying performance bottlenecks. We derived the pattern of content delivery that can incorporate swarming in order to effectively utilize the outgoing bandwidth of participating peers and thus minimize the content bottleneck in the system. This in turn led us to the desired block scheduling scheme at individual peers. Through extensive packet-level simulations, we examined the effect of key factors on PRIME performance. Our simulations have shown that in designing a mesh-based P2P streaming system the most important factors affecting the performance are the ratio of bandwidth to peer degree, peer degree, and the amount of buffer size. Heterogeneity and asymmetry of peers' bandwidth, presence of free riders and peer dynamics have minor effect on the performance as long as there are sufficient amount of resources in the system. In particular, Some of our main findings can be summarized as follows:

- Ensuring the same ratio of bandwidth to degree among participating peers minimizes the bandwidth bottleneck in the overlay.

- There is a sweet range for peer degree over which swarming content delivery exhibits a good performance and effectively scales with peer population. The lower bound of this range is 6 but the upper bound is determined by peer bandwidth.

- The minimum buffer requirement at each peer is directly proportional to the total duration of the diffusion and swarming phases for each block. The minimum duration of diffusion phase depends on the depth of the overlay whereas the minimum duration of swarming phase depends on the connectivity of the overlay.

Bi-directional overlays require larger buffering at individual peers due to the lower diversity in connectivity which adversely affects swarming content delivery.

- In a properly connected overlay with the sufficient amount of resources (*i.e.*, aggregate outgoing bandwidth among peers is not smaller than their aggregate incoming bandwidth) neither the heterogeneity and asymmetry of access link bandwidth nor the location of high bandwidth peers significantly affects the delivered quality to individual peers. However, the presence of free-riders may limit the connectivity between regions of the overlay and thus prevent the delivery of a subset of blocks to some regions of the overlay.

- The block scheduling scheme at individual peers should pull any newly generated blocks (with the highest timestamps) from parents to ensure proper diffusion of content through the overlay. Besides this requirement, the actual criteria for selecting blocks from individual parents does not have a significant impact on the performance of content delivery as long as load is properly balanced among parents.

- Incorporating some light weight coordination mechanism (*i.e.*, careful block swapping and loss detection) at source can significantly improve overall performance of content delivery.

- The more imbalanced the bandwidth-degree ratio among participating peers (*i.e.*, the more distorted the overlay) becomes, the lower the diffusion rate of new blocks through the overlay becomes, and the lower the delivered quality to individual peers would be.

Overall, our study of mesh-based P2P streaming systems reveal that with careful attention to the design choices, a mesh-based system can indeed deliver a good performance for scalable live P2P streaming under various network settings.

The natural next step in our research is comparing the two existing approaches on live P2P streaming. Towards that, in the next chapter, we present our work on the comparing the performance of the PRIME as a representative mesh-based approach with the traditional approach on live P2P streaming, namely, tree-based.

CHAPTER IV

COMPARATIVE STUDY ON TREE- AND MESH-BASED P2P STREAMING APPROACHES

Material in this chapter was adapted from a paper [5] previously published in *IEEE Infocom*, 2007, and co-authored by Prof. Reza Rejaie and Dr. Yang Guo. The experimental work is entirely mine. The text is written jointly by myself and Prof. Reza Rejaie. The co-authors provide guidance on the technical part.

As we have discussed in Chapter II, existing approaches for live P2P streaming can be generally divided into two classes: *tree-based* approaches and *mesh-based* approaches. The tree-based P2P streaming approach expands on the idea of end-system multicast [18] by organizing participating peers into multiple diverse trees. Then, each description of a Multiple Description Coded (MDC) content is pushed through a separate tree [29, 56]. The mesh-based approach has been discussed in Chapter III in details. Most of the previous studies on P2P streaming have focused on a particular mechanism and evaluated certain aspects of its performance. However, to our knowledge, the performance of these two classes of P2P streaming approaches have not been directly compared.

4.1. Contributions

In this chapter, our goal is to compare and contrast the performance of tree-based and mesh-based P2P streaming approaches. We provide an overview of a representative protocol in each class and expose their similarities and differences. We then compare the performance of tree- and mesh-based approaches using the representative protocols in two steps as follows:

First, we examine the performance of content delivery in these approaches over a properly connected and static overlay. We present the notion of "delivery tree" for individual blocks in the mesh-based approach which enables us to clearly compare the behavior of content delivery in tree- and mesh-based approaches. We also examine the effect of peer degree (*i.e.*, number of trees), bandwidth heterogeneity, and peer population. Our evaluations reveal that swarming content delivery in mesh-based approach exhibits a superior performance across a wide range of scenarios.

Second, we investigate the ability of both approaches to cope with churn from two angles *(i)* The performance of content delivery on a distorted overlay, and *(ii)* The cohesion of the overlay structure under persistent churn.

We model a distorted overlay by removing a random subset of participating peers from a properly connected overlay without repairing it. We show that the swarming delivery in the mesh-based approach can effectively utilize available resources over distorted overlays whereas the tree-based approach exhibits poor performance in such circumstances. We also quantify the cohesion of the overlay under churn using three metrics: ancestor changing rate, the average degree of connectivity, and the frequency of deadlock events (only in the tree-based approach). Our results indicate that peers always experience a higher degree of stability in the mesh-based approach.

Throughout this chapter, we only focus on a scenario where there is a balance between supply and demand for resources (namely bandwidth) in the overlay, *i.e.*, the aggregate incoming and outgoing bandwidth across all peers are equal. Clearly, the performance of any P2P streaming approach is affected by the availability of resources in the system. However, my goal is to examine the performance of tree- and mesh-based approaches when participating peers are willing to contribute as much resource as they consume. Furthermore, this appears to be the basic scenario to conduct such a comparison.

In summary, we make two important contributions in this chapter:

- Leveraging the notion of delivery tree for individual blocks, we identify the key differences between mesh-based and tree-based approaches to P2P streaming. This in turn sheds an insightful light on the inherent limitations and potentials of these two approaches.

- We identify the underlying causes for the observed differences between tree- and mesh-based approaches.

4.2. Background

In this section, we present an overview of multiple-tree-based P2P streaming approach. The mesh-based approach that we employ for the comparison is similar to PRIME which has been described in Chapter III and has been shown to outperform other mesh-based solutions [69, 127].

We assume all pairwise connections for data delivery between peers are congestion controlled in both tree- and mesh-based approaches. This ensures that these approaches behave in a network-friendly fashion and achieve proper bandwidth sharing among incoming (and outgoing) connections to (from) individual peers. We also assume that both approaches leverage *Multiple Description Coding (MDC)* to accommodate the bandwidth heterogeneity among participating peers. As noted in Section ch:rel:sec:cod in MDC, a stream is encoded into multiple sub-streams called *description*. Each description can be independently decoded. Furthermore, receiving multiple unique descriptions results in a higher quality. This enables individual peers to receive the proper number of descriptions proportional to their aggregate incoming bandwidth in order to maximize their received quality.

4.2.1. Multiple-tree-based Approach

In the tree-based approach, an overlay construction mechanism organizes participating peers into multiple trees. Each peer determines a proper number of trees to join based on its access link bandwidth. To minimize the effect of churn and effectively utilize available resources in the system, participating peers are organized into multiple *diverse* trees. Toward this end, each peer is placed as an *internal* node in only one tree, and as an *external* (or leaf) node in other trees. Then, each description of an MDC encoded content is delivered through a specific tree. The content delivery is a simple push mechanism where internal nodes in each tree simply forward any received blocks for the corresponding description to all of their child nodes. Therefore, the main component of the tree-based P2P streaming approach is the tree construction algorithm.

Tree Construction Algorithm: The goal of the tree construction is to maintain multiple balanced, stable and short trees. In this chapter, we use the following central tree construction algorithm that to our knowledge, represents the best practice among existing solutions [29, 56] and discussed in Subsection 2.2.2.:

Each peer is placed as an internal node in only one tree and leaf node in other participating trees. When a peer joins the system, it contacts the bootstrapping node to identify a parent in the desired number of trees. To keep the population of internal nodes balanced among different trees, a new node is added as an internal node to the tree that has the minimum number of internal nodes. To maintain short trees, a new internal node is placed as a child for the node with the lowest depth (the first node as we traverse the tree in a breadth-first fashion) that can accommodate a new child or has a child that is a leaf. In the latter case, the new node replaces the leaf node and the partitioned leaf should rejoin the tree similar to a new leaf. When an internal node of a tree departs, each one of its child nodes as well as the subtree rooted at them are partitioned from the original tree, and thus should rejoin the tree. Peers in such a partitioned subtree initially wait for the root of the subtree to rejoin the tree as an internal node. If the root is unable to join the subtree after a certain period of time, individual peers in a partitioned subtree independently rejoin the tree with the same position (as leaf or internal node). A tree can always accept a new internal node. However, in the presence of churn, a tree could become *saturated* and thus unable to accept any new leaf node. We denote this as a *deadlock event*. A deadlock event occurs when a tree loses a fraction of its internal nodes within a short period of time which reduces the number of leaf nodes that it can accommodate. In such a scenario, the number of internal nodes at different trees becomes imbalanced, where spare slots for leaf nodes are available on other trees but they can not be used to resolve the deadlock of the saturated tree. When a leaf node experiences deadlock, it periodically tries to rejoin the tree until it succeeds.

4.3. Similarities and Differences Between Tree and Mesh-based Approaches

In this subsection, we describe the similarities and differences between two approaches which helps us identify the underlying causes for the observed behavior by each approach in our evaluations.

Similarities: The tree-based and mesh-based approaches have a great deal of similarities as follows: First, while these approaches use different overlay construction algorithms, the overall shape of their resulting overlays is very similar. More specifically, the superimposed view of multiple diverse trees is in fact the same as a directed random mesh. Second, the content delivery in both approaches enable individual peers to receive different pieces of the content. At the peer level, each peer receives content from multiple parents and sends content to multiple child peers in both approaches. At the system level, the collection of edges used for the delivery of a single block form source to all participating peers form a source-rooted tree in both approaches that we call the *delivery tree.* Third, both approaches require participating peers to maintain a loosely synchronized playout time that is sufficiently (τ seconds) behind source's playout time. This requires τ seconds worth of buffering at each peer which accommodates the diversity of different paths from source in the tree-based approach, and out-of-order block arrival in the swarming content delivery of the mesh-based approach. The value of τ depends on the maximum hop count from source to different participating peers through the overlay which is a function of peer population and peer degree. For a fair comparison, we assume that both approaches use the same value of τ in comparable scenarios.

Differences: The key difference between the mesh-based and the tree-based approaches is how the delivery tree of individual block is formed. In the tree-based approach, the delivery tree for all blocks of a particular description is the corresponding overlay tree for that description. In essence, the delivery tree of each block is indeed pinned down by the tree construction mechanism because of the *static* mapping of descriptions to trees. This has an important implication: when the bandwidth of a connection is less than the description bandwidth, the blocks for that description can not be "streamed" at a proper rate to all the descendant peers. In contrast, in the mesh-based approach, the delivery tree for individual blocks is dynamically shaped as the block traverses through the overlay. The dynamic formation of the delivery tree enables the mesh-based approach to effectively utilize the available resources. In particular, when a connection has low bandwidth, its descendant peers can still receive their required blocks through alternative paths from other parents. The dynamic formation of the delivery tree in the mesh-based approach is essential in understanding its behavior, and it is explained in further details in the following subsection.

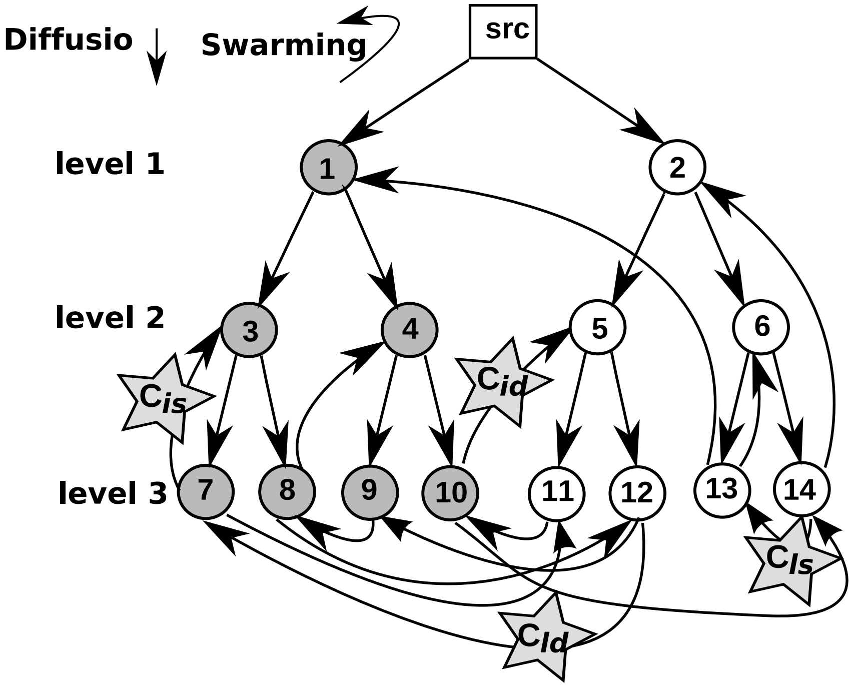


FIGURE 22.: Organized view of a random mesh overlay, along with the various types of swarming connections

4.3.1. Delivery Trees in Mesh-based Approach

To derive the delivery tree in the mesh-based approach, we need to present the proper pattern of content delivery over a mesh that maximizes the utilization of outgoing bandwidth among participating peers. Toward this end, we utilize the notion of the organized view of a randomly connected mesh that is introduced in Subsection 3.4.2.. Recall that, in an organized view of a random mesh, we group peers into *levels* based on their shortest distance (in hops) from source through the overlay as shown in Figure 22.. Peers that are one-hop away from source (source's children) are in level 1, peers that are two hops away from source are in level 2, and so on. The number of levels is equal to the *depth* of the overlay or the maximum distance of a peer from source. To efficiently utilize source's bandwidth, we assume that source should deliver each block only once.

The pattern of delivery for a single block over an organized mesh should consist of the diffusion and swarming phases in order to maximize the utilization of outgoing bandwidth among participating peers as described in Subsection 3.4.3..

In the diffusion phase, once a new block becomes available at the source, a single peer p in level 1 pulls the block during the next interval Δ. Then, all the p's child peers in level 2 pull a copy of the block in the following interval and so on. All connections from peers in level i to peers in level $i + 1$ (i<*depth*) are used for diffusing new blocks through the and called diffusion connections. The diffusion connections are shown with straight arrows in Figure 22.. Recall that, since each block is only delivered once from the source, the subset of peers that receive a block during its diffusion phase, form a subtree, called *diffusion subtree*. The diffusion subtree consists of a peer in level 1 (as its root) and all of its descendant peers in lower levels. For example, the shaded nodes in Figure 22. form a single diffusion subtree. Note that the number of distinct diffusion subtrees in the overlay is equal to the number of peers in level 1 (*e.g.*, two diffusion subtrees in Figure 22.).

During the swarming phase, peers on different diffusion subtrees exchange their new blocks (or swarm) to contribute their outgoing bandwidth. All the connections from a peer in level i to a peer in level j ($j \leq i$) are used for swarming and thus called swarming connections. These connections are shown with curly arrows in Figure 22..

The swarming connections can be divided into the following four groups based on the locations of two peers that they connect:

- Connecting peers at the bottom of two different diffusion subtrees (C_{ld}),
- Connecting peers at the bottom of the same diffusion subtree (C_{ls}),
- Connecting a peer at the bottom of one diffusion subtree to an internal peer on a different diffusion subtree (C_{id}),
- connecting a peer at the bottom of one diffusion subtree to an internal peer on the same diffusion subtree (C_{is}).

A sample connection from each group is marked with star and proper label in Figure 22.. The delivery tree of a block in the mesh-based approach consists of two parts:

- *the top portion* of the delivery tree that must be the same as one of the diffusion subtrees,
- *the bottom portion* of the delivery tree consists of a collection of swarming connections that are extending (or hanging from) the diffusion subtree.

Our block scheduling algorithm implies that different groups of swarming connections can only be attached to the delivery tree at certain locations based on the following rules:

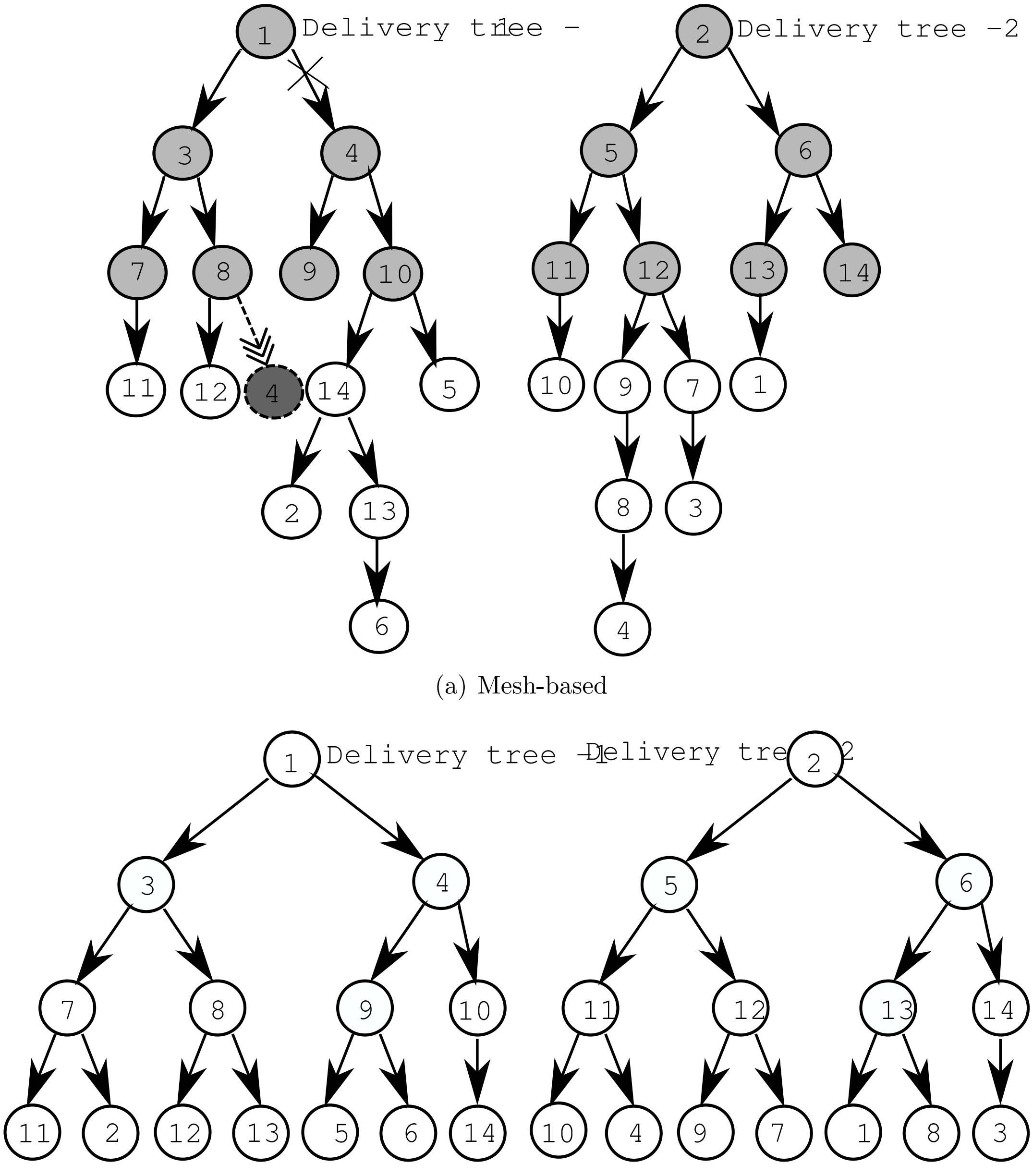
- C_{is} and C_{ls} can be attached at any part of the bottom portion of the delivery tree. These connections enable participating peers to receive any missing blocks through swarming parents and cope with a diffusion connection with low bandwidth.

- C_{ls} and C_{ld} can only be attached to C_{ls} or C_{is} type connections. Otherwise, they form an ending branch for the delivery tree.

- C_{id} and C_{ld} can only be attached to the diffusion subtree.

- C_{is} and C_{id} can only be attached as an ending branch of the delivery tree.

Figures 23.(a) and 23.(b) illustrate two delivery trees in the mesh-based and tree-based approaches for the overlay in Figure 22., respectively. We summarize our main points in this section as follows: The delivery tree of individual blocks in the tree-based approach is determined by the overlay construction mechanism. As a result, a low bandwidth connection in an overlay tree can limit the rate of data delivery to all of the downstream peers. In contrast, the delivery tree in the mesh-based approach is dynamically determined by the collective behavior of block scheduling mechanisms among participating peers (*i.e.*, swarming content delivery). This enables individual peers to gracefully cope with a low bandwidth connection by receiving their desired blocks from other parents through other paths. For example, if the connection from peer 1 to peer 4 in Figure 23.(a) has a low bandwidth, peer 4 (as well as its descendant peers in the diffusion subtree, such as peers 9 and 10) can still receive a subset of blocks from other swarming parents (*e.g.*, peer 8). In essence, the



(a) Mesh-based

(b) Tree-based

FIGURE 23.: Delivery trees for mesh- and tree-based approaches.

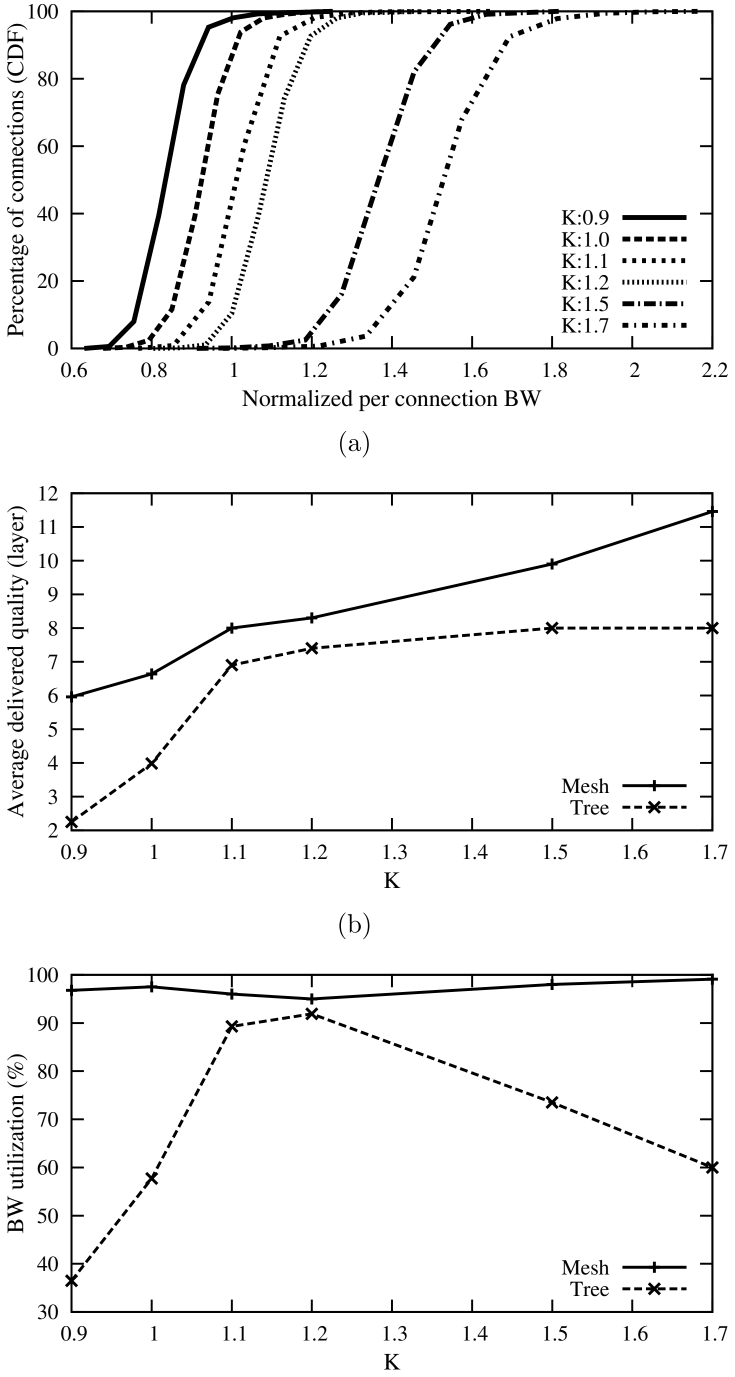
dynamic formation of a delivery tree implies that a peer can appear at different parts of the delivery tree for different blocks. One side effect of the dynamic formation of the delivery tree in the mesh-based approach is their longer depth compared to the mesh-based approach as shown in Figures 23.(a) and 23.(b).

4.4. Performance Comparison: Static Group

In this section, we examine the performance of content delivery mechanism in both approaches over a static overlay using *ns* simulations. In our simulations, the physical topology is generated using Brite [126] with 15 AS, 10 routers per AS in top-down mode, and *RED* queue management at all routers. The delay on the access links is randomly selected between [5ms, 25ms]. Results are averaged across multiple simulations with different random seeds. All pairwise connections between peers employ *RAP* congestion control mechanism [124]. Core links have high bandwidth (4Gbps to 10Gbps) and thus individual connections only experience bottlenecks at the edge. To quantify the utilization of available bandwidth for each connection, we use the following methodology to decouple the available bandwidth from the available content as follows: when a parent experiences content bottleneck and does not have any useful block to deliver to a particular child, it sends a especially marked block with the same size to that child. The *bandwidth utilization* is defined as the ratio of the number of data blocks to the total number of delivered blocks. We also define *the average delivered quality* for each peer as the average number of descriptions it receives during a session.

In both approaches, all peers maintain synchronized playout time that is τ seconds behind the source's playout time. To model a live streaming session, each peer starts playing the content τ seconds after simulation starts, and maintains τ seconds worth of content. The value of τ is selected to be the minimum value that can accommodate in-time delivery of blocks for a given population and peer degree. Based on this strategy, we conservatively set τ to 24 seconds in our simulations. We use the following default values for other parameters: each stream has 20 descriptions and all descriptions have the same constant bit rate of 80Kbps (bw_d). Δ is set to 4 seconds. Each scenario consists of 200 homogeneous peers with symmetric bandwidth, and access link bandwidth of all peers is set to $deg \cdot bw_d$ where deg denotes the degree of each peer. Thus, each peer should be able to receive deg descriptions which we refer to as *target quality*.

In both approaches, the source degree is equal to the peer degree (deg). Furthermore, source bandwidth is set to the minimum value that is required for the delivery of the desired aggregate quality to the overlay (*i.e.*, the delivered quality to all peers in level 1, collectively). The aggregate delivered quality in each simulation is equal to the quality that peers with the highest bandwidth can obtain.



(a)

(b)

(c)

FIGURE 24.: Effect of per-connection bandwidth: (a) Distribution of normalized per-connection bandwidth. (b) and (c) Average delivered quality and bandwidth utilization for various values of K for both mesh and tree, respectively.

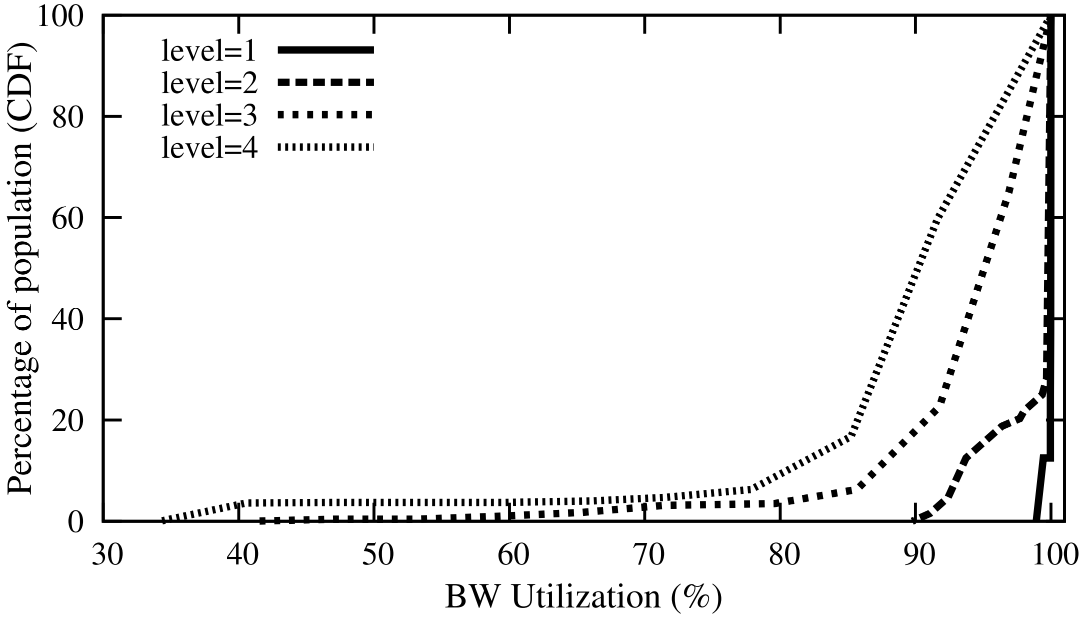
4.4.1. Per-connection Bandwidth

We first examine the effect of per-connection bandwidth on the system performance. Since all peers have the same incoming and outgoing degree of deg, by setting the access link bandwidth to $deg \cdot K \cdot bw_d$, we can control the average per-connection bandwidth to be $K \cdot bw_d$. We can vary the access link bandwidth by changing K in order to investigate the effect of per-connection bandwidth on system performance. Figure 24.(a) depicts the distribution of per-connection average bandwidth (normalized by bw_d) for different values of K where peer degree is 8. This figure clearly demonstrates that different connections obtain different average bandwidth due to the dynamics of congestion control. As the peer bandwidth increases, the median value of the distribution proportionally increases and it becomes slightly more skewed. The key question is *"whether the distribution of per-connection bandwidth affects the performance of tree- or mesh-based P2P streaming approach?"*

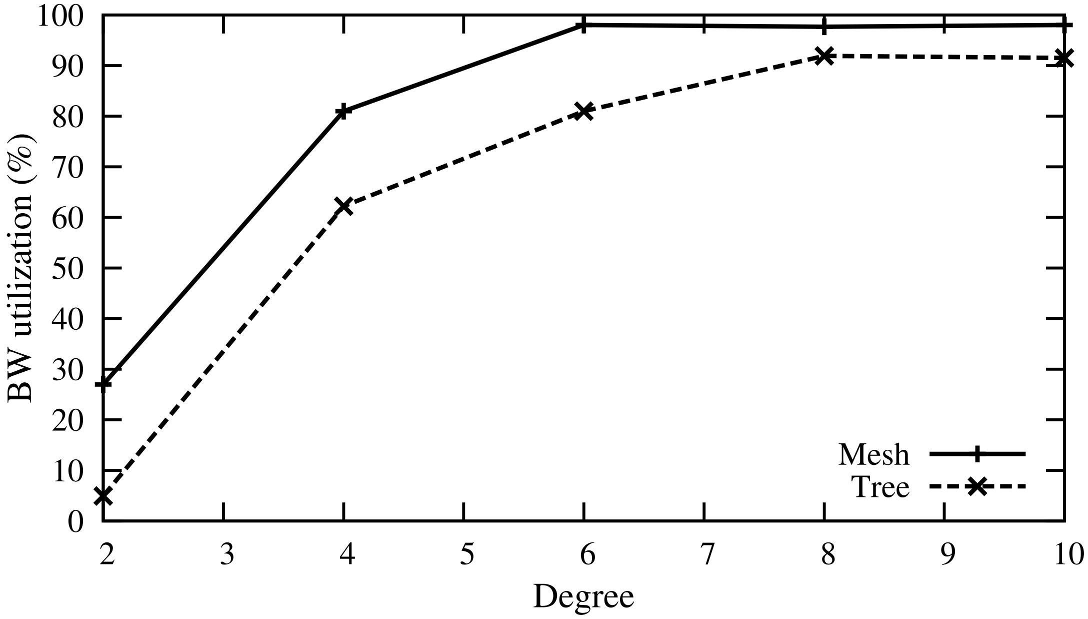
Figure 24.(b) presents the average delivered quality as a function of K in both approaches. Figure 24.(b) reveals that the average delivered quality in the mesh-based approach is proportionally improved with the peer bandwidth and *can* even exceed the target quality. In contrast, the average delivered quality in the tree-based approach is poor when connection bandwidth is less than or equal to the description bandwidth ($K \leq 1$). As the per-connection bandwidth increases, the average delivered quality reaches the targeted quality of deg descriptions but cannot go beyond this limit regardless of the per-connection bandwidth.

Figure 24.(c) shows the average bandwidth utilization across all connections as a function of K. In the mesh-based approach, participating peers achieve high bandwidth utilization (>%95) and can properly adjust the delivered quality for any value of per-connection bandwidth. In contrast, the aggregate bandwidth utilization in the tree-based approach has a sweet spot (at K=1.2) where it reaches %90. However, for all other values of K, it exhibits a significantly lower bandwidth utilization. The poor bandwidth utilization for small values of K is due to extended effect of a single low bandwidth connection on all of its downstream connections. But when the per-connection bandwidth is large, the bandwidth of individual connections significantly exceeds the description bandwidth. This results in the *content bottleneck* since parent peers do not have sufficient useful content to utilize the available bandwidth. *In summary, the tree-based approach has a sweet spot for the ratio of per-connection bandwidth to description bandwidth where high resource utilization and thus high delivered quality is achieved. In contrast, the mesh-based approach can effectively utilize any value of peer bandwidth and deliver a proportionally higher stream quality.*

For the remaining evaluations in this section, we set the value of K to 1.2 for the tree-based approach to achieve its best performance. In a nutshell, this implies that our results in this section represent an upper bound for the performance of the tree-based approach.

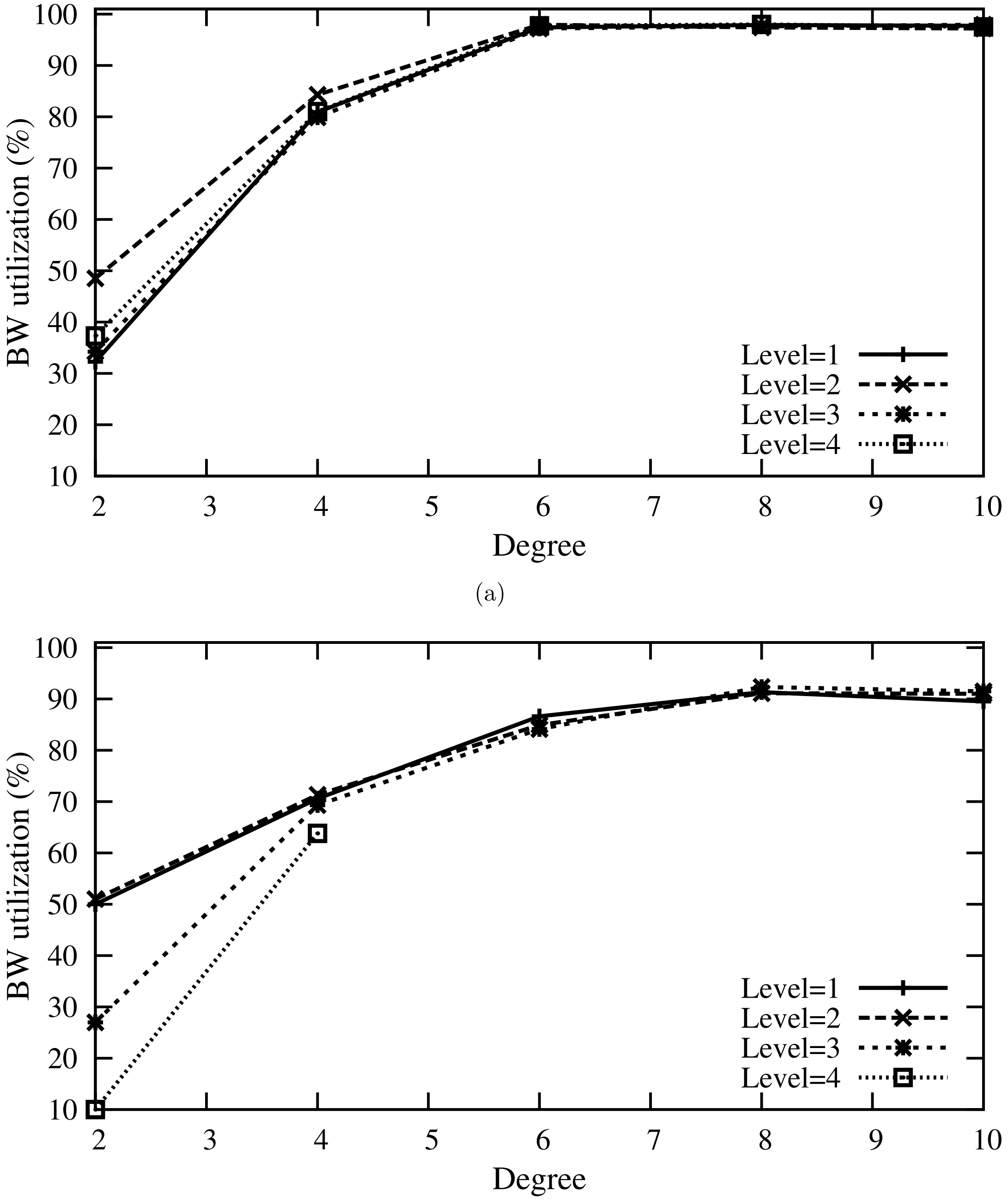


(a)



(b)

FIGURE 25.: Effect of peer degree: (a) Distribution of per-connection bandwidth utilization for peers in various levels for tree-based approach. (b) Average bandwidth utilization for tree-based and mesh-based approaches as a function of peer degree.



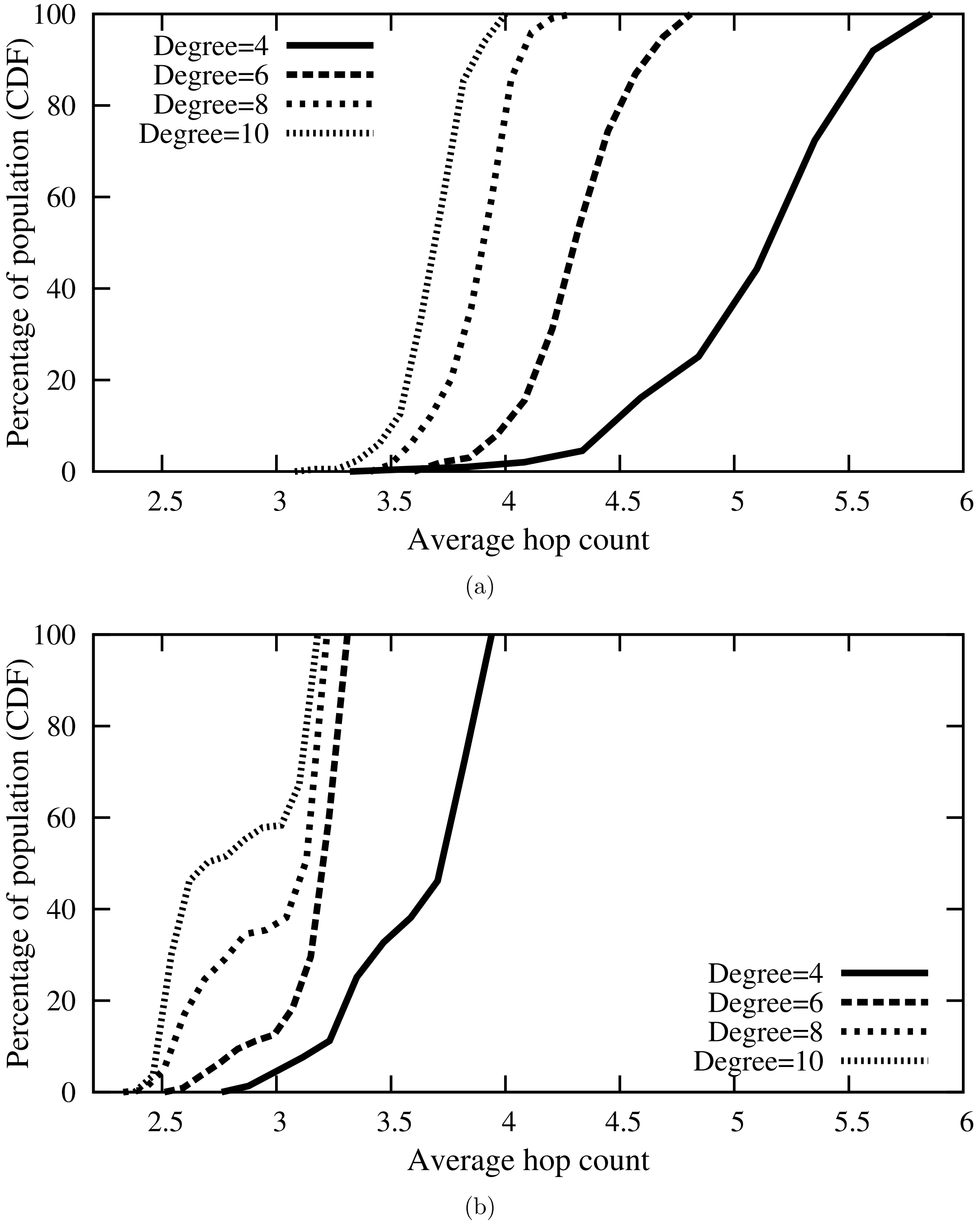
(a)

(b)

FIGURE 26.: Effect of peer degree: (a) and (b) Average bandwidth utilization among peers at various distance from source in mesh- and tree-based approach, respectively.

4.4.2. Peer Degree or Number of Trees

In this subsection, we investigate the effect of peer degree on system performance. Figure 25.(a) shows the distribution of per-connection bandwidth utilization across peers that are n hops away from source and their child peers (labeled as level n) for different values of n in the tree-based approach. This figure demonstrates that connections that are further away from the source have a lower average utilization due to the higher probability of experiencing low bandwidth among their upstream connections. The mesh-based approach exhibits a high bandwidth utilization (>%95) across all connections in a similar setting since it can cope with content bottlenecks (the result is not shown here). Figure 25.(b) depicts the average bandwidth utilization as a function of peer degree for both approaches. This figure reveals that by increasing peer degree the bandwidth utilization rapidly improves for both approaches. In the tree-based approach, increasing peer degree reduces the depth of all trees which decreases the number of upstream connections and thus the probability of a content bottleneck. In the mesh-based approach, the improved utilization for higher degree is due to the larger number of parents peers which provides more flexibility for block scheduling and reduces the probability of content bottleneck. More importantly, the mesh-based approach exhibits a higher utilization across all degrees primarily due to its flexibility to dynamically map its required content to parents and effectively use their available bandwidth.



(a)

(b)

FIGURE 27.: Effect of peer degree: (a) and (b) Distribution of average hop count among peers for mesh- and tree-based approaches, respectively.

Figure 26.(a) and 26.(b) show the average bandwidth utilization among peers at different distance from source. These figures reveal that the aggregate bandwidth utilization does not depend on peers location in the overlay for both approaches. The aggregate average bandwidth utilization depends on the average distance of each peer across the delivery tree of different blocks. In the mesh-based approach, because of the random connectivity among peers, average distance of all peers is very similar. In the tree-based approach, for large peer degrees, the average distance of all peers is similar. To explain this, we note that the average distance of each peer is primarily determined by the depth of individual trees since each peer is placed as a leaf in all but one tree. On the other hand, the observed disparity for small peer degrees (degree < 4) in the tree-based approach is due to the pronounced effect of peer distance on the tree where it serves as an internal node.

Average hop count (*i.e.*, the number of peers that a block visits before reaching each peer) among delivered blocks to each peer represents its average distance across all delivery trees. Figure 27.(a) and 27.(b) depict the distribution of average hop count among delivered blocks to each peer. These figures present two interesting points:

- The average path length is generally longer in the mesh-based approach. This is mainly due to the flexibility of swarming delivery that allows a peer to receive missing blocks through a longer path from its swarming parents.

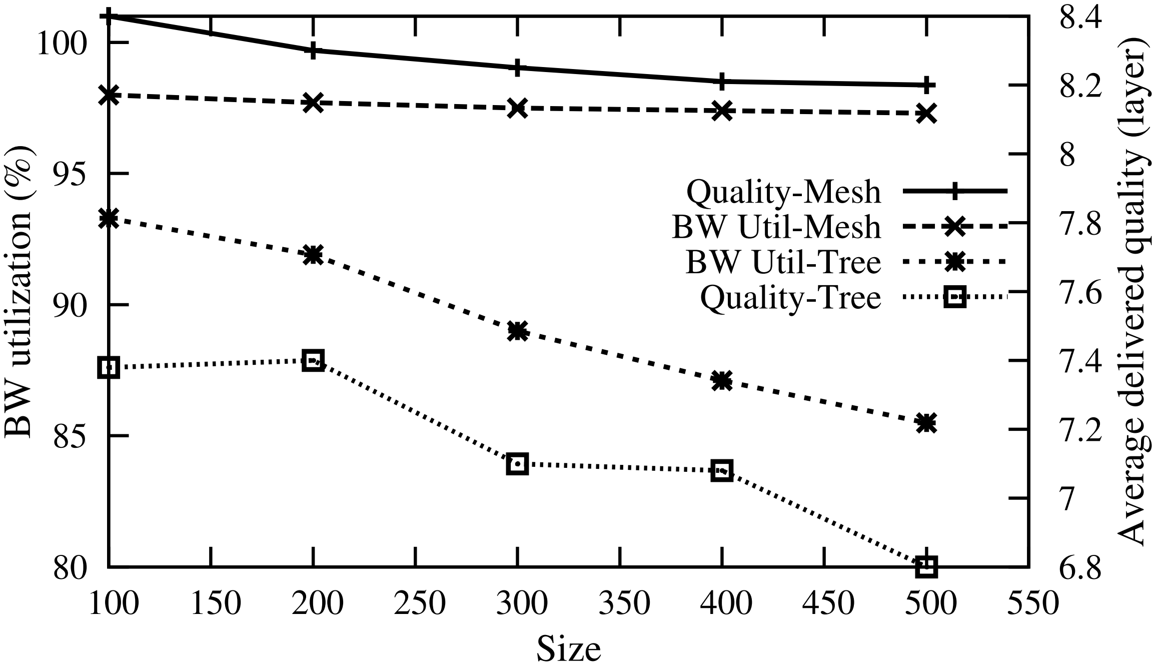


FIGURE 28.: Effect of peer population: Bandwidth utilization and average delivered quality for both mesh and tree-based approaches.

- As the peer degree increases, the average path length in both approaches decreases but for different reasons. For the tree-based approach, increasing degree reduces the depth of all trees and results in a lower average hop count for individual peers.

In the mesh-based approach, increasing degree reduces the depth of delivery trees by providing more shortcuts in the mesh. This in turn enables each peer to receive more blocks through a shorter path which leads to a shorter and more homogeneous average path length among peers.

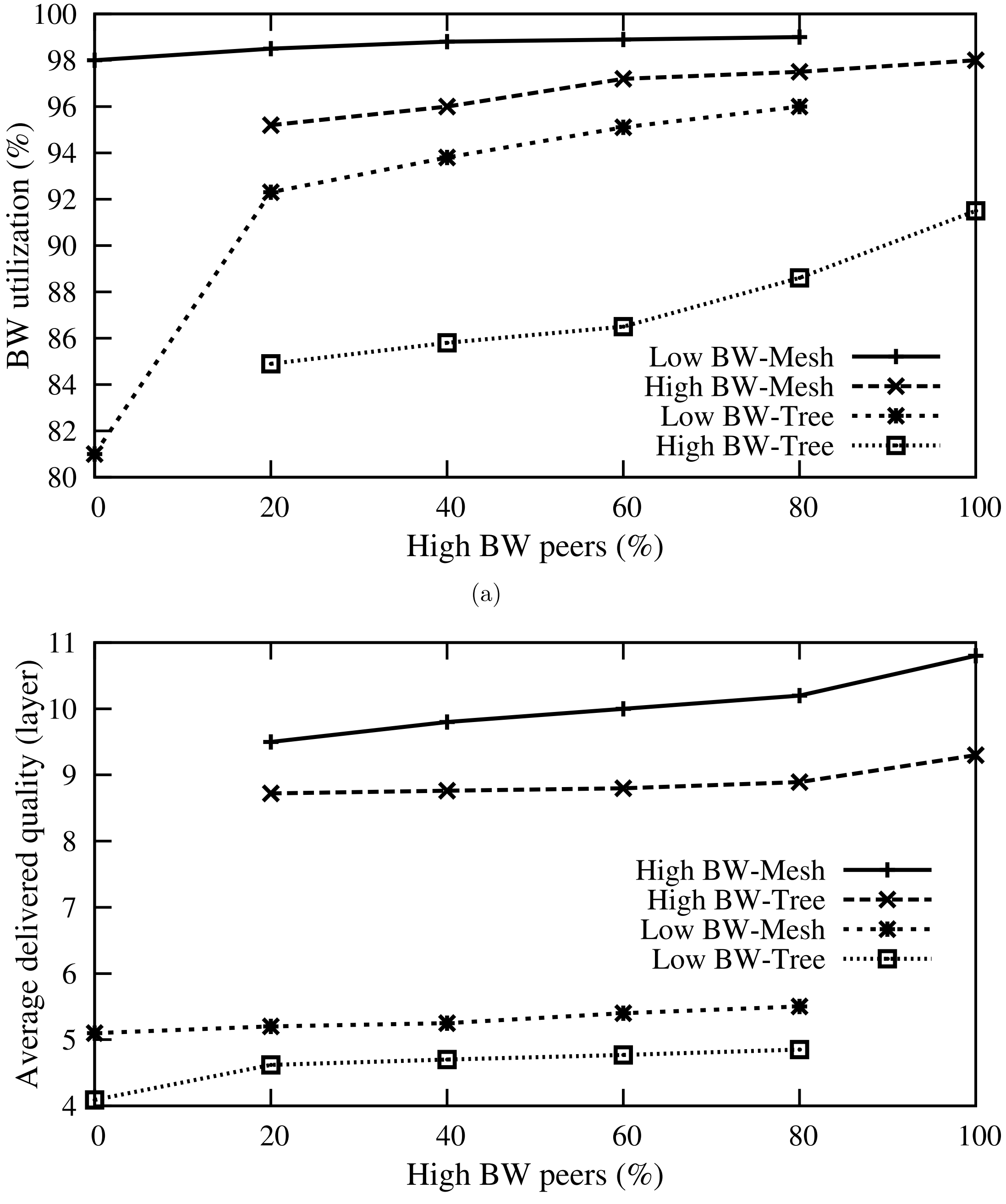
4.4.3. Peer Population

Another key issue is "how the performance of these approaches is affected by group size?". To investigate the effect of group size, we examine a group of peers with homogeneous bandwidth and peer degree 8. Figure 28. presents the average bandwidth utilization and average delivered quality among all peers for both approaches as a function of group size.

Figure 28. reveals that as the group size increases, both the utilization and the delivered quality in the tree-based approach gradually drops whereas the mesh-based approach consistently exhibits high performance. To explain the behavior of the tree-based approach, we note that for a given peer degree, the depth of individual trees increases with the group size. This in turn decreases the per-connection bandwidth utilization due to the higher chance for content bottleneck among different connections. In contrast, the flexibility of swarming content delivery enables mesh-based approach to effectively scale with group size.

4.4.4. Bandwidth Heterogeneity

To explore the effect of bandwidth heterogeneity among participating peers, we consider two groups of peers with symmetric bandwidth of 480 Kbps and 960 Kbps, and peer degree of 5 and 10, respectively. Since the bandwidth to degree ratio is the same for both groups, the average per-connection bandwidth should be roughly the same across all connections. Figure 29.(a) depicts the average bandwidth



(a)

(b)

FIGURE 29.: Effect of bandwidth heterogeneity: (a) and (b) Percentage of bandwidth utilization and average delivered quality for high and low bandwidth peers as a function of the percentage of high bandwidth peers for both mesh and tree-based approaches, respectively.

utilization for high and low bandwidth peers in both approaches as a function of the percentage of high bandwidth peers in the group. Figure 29.(b) presents the average delivered quality in the same scenarios. These figures indicate that both groups of peers consistently achieve a higher utilization and receive a significantly better quality in the mesh-based approach. The bandwidth utilization and thus the delivered quality to individual peers in the mesh-based approach depends on the aggregate quality of available content among their parents as discussed in Subsection 3.7.1.. Therefore, as the percentage of high bandwidth peers increases, the performance of the mesh-based approach gradually improves. In the context of tree-based approach, the main determining factor for both utilization and quality is the average depth across different trees. Increasing the percentage of high bandwidth peers rapidly drops depth of all trees which in turn improves both utilization and the delivered quality.

4.5. Performance Comparison: Dynamic Group

The dynamics of peer participation (or churn) could disrupt content delivery and adversely affect the delivered quality to participating peers. Such a disruption occurs when a peer loses its direct parent, or any upstream node along the path from source. In this section, we examine the effect of churn on the tree- and mesh-based approaches.

To cope with churn, an affected peer should rejoin the proper tree in the tree-based approach, or connect to a new parent in the mesh-based approach. While the effect of churn is often transient, its impact on the delivered quality depends on many factors including details of the recovery mechanism, amount of buffering at each peer, and the characteristics of churn. Therefore, instead of quantifying the delivered quality in dynamic scenarios, we examine the performance of these approaches at the following two levels:

- The performance of content delivery on distorted overlays, and
- the cohesion of the overlay structure under persistent churn.

This methodology not only allows us to separately examine the effect of churn on content delivery and overlay construction mechanisms but also simplifies the comparison between two candidate approaches.

4.5.1. Content Delivery in Distorted Overlay

To model a distorted overlay, we use a properly connected overlay in both tree- and mesh-based approaches and then assume that $x\%$ of randomly selected peers simultaneously depart without repairing the overlay. The resulting distorted overlay represents the worst case scenario for the overlay as it evolves due to churn. By changing x, we can control the level of distortion in the overlay. Figure 30. depicts the median utilization of aggregate bandwidth among peers in a distorted overlay (as well as its 5th and 95th percentile as a bar) for both approaches as a function of x. This figure clearly illustrates that bandwidth utilization among peers in the mesh-based approach is significantly higher than the tree-based approach. This is primarily due to the ability of the swarming delivery to cope with unbalanced incoming/outgoing degree among participating peers in a distorted overlay.

In contrast, the diverse nature of tree structures implies that the departure of any peer in the tree-based approach reduces the delivered quality to all of its descendant peers on the tree where it serves as an internal node. Therefore, the departure of a larger fraction of peers leads to a proportionally larger drop in bandwidth utilization and widens its distribution among peers.

4.5.2. Cohesion of the Overlay Under Churn

We now turn our attention to the ability of each approach to maintain a cohesive overlay in the presence of churn. For this analysis, we ignore content delivery and focus on the following key aspects of overlay dynamics in both approaches:

- The frequency of change among ancestor nodes,
- The frequency of deadlock events,
- The average connectivity for individual peers.

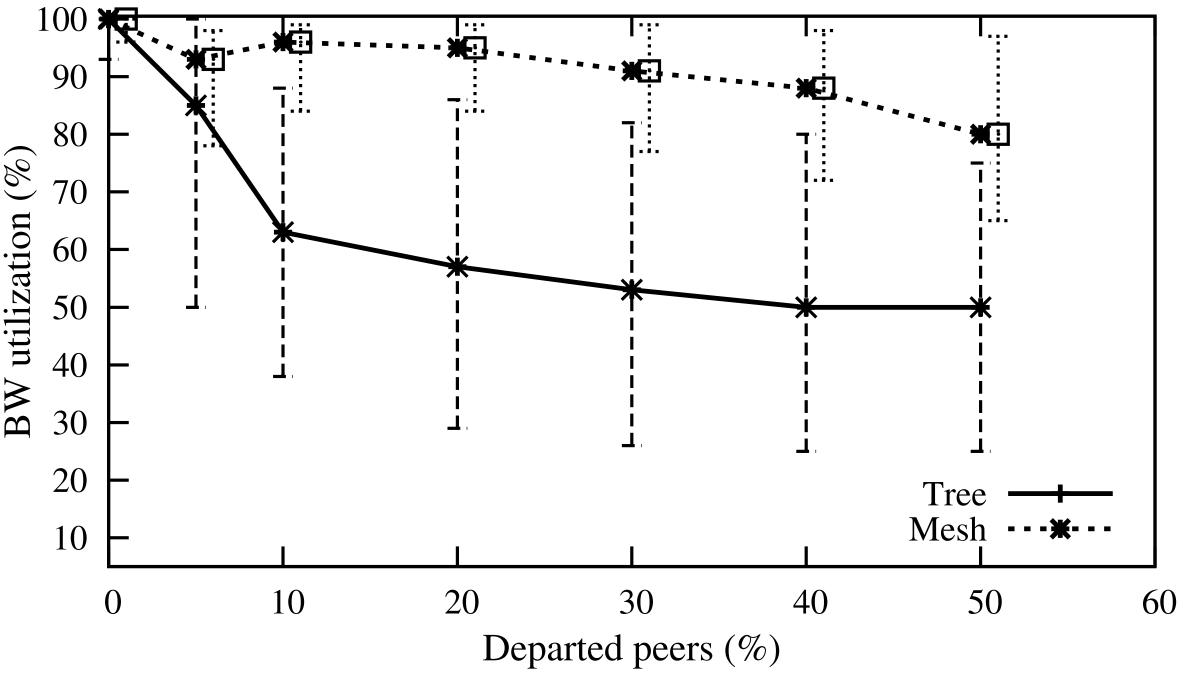
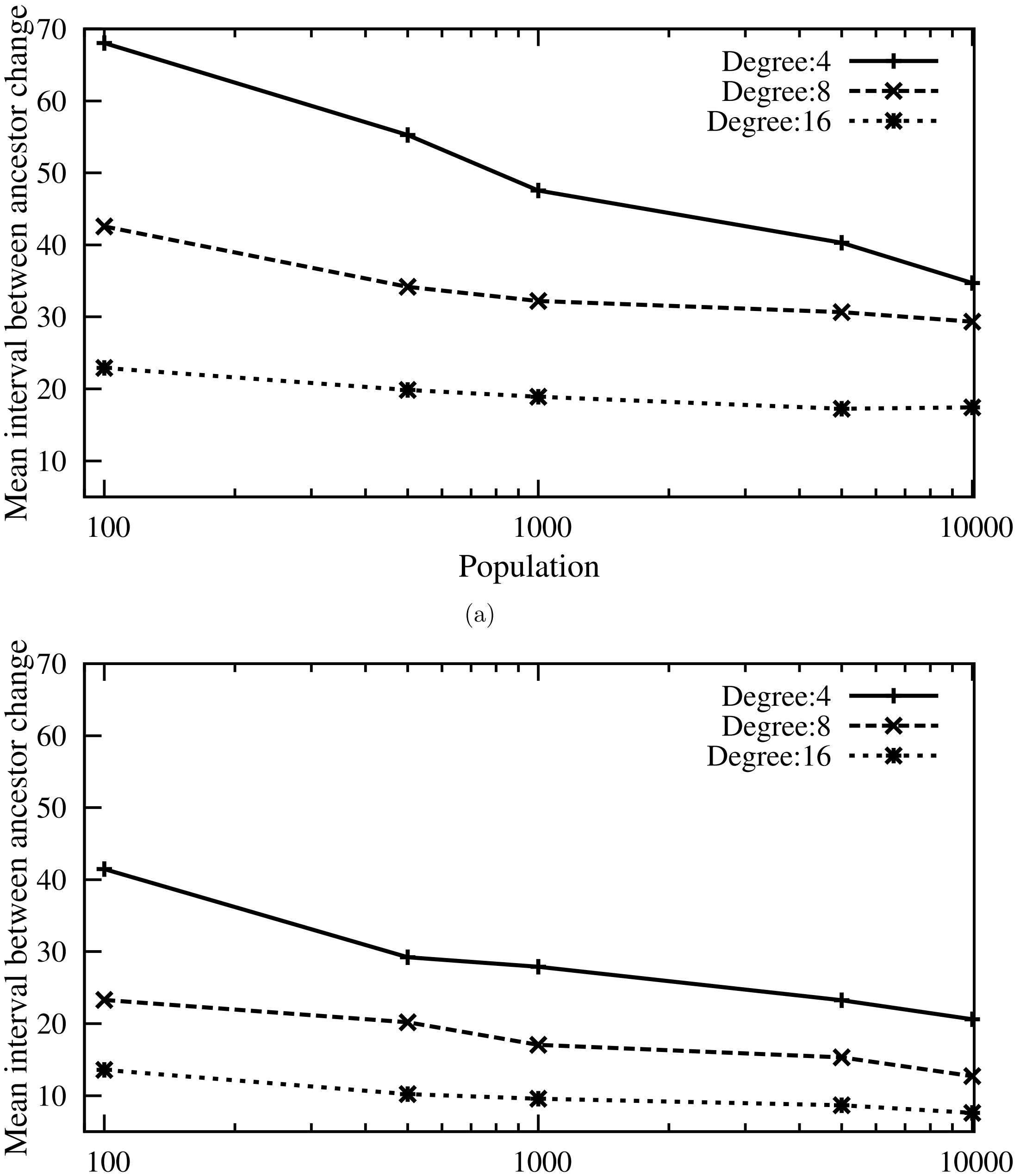


FIGURE 30.: Effect of distorted overlays: Median, 5th and 95th percentile of bandwidth utilization among peers after x% of randomly selected peers have departed.

For this analysis, we use our session level P2P simulator, called *psim*. *psim* abstracts out packet level dynamics and allows us to examine significantly larger group sizes. Furthermore, *psim* enables us to accurately model churn by directly controlling the distribution of session length of each peer. It simulates the pairwise latency between peers using the King dataset [128]. *psim* also uses a central bootstrap mechanism with a random selection algorithm for peer discovery and peer selection. To incorporate a realistic model for churn in our simulations, we select peer session times from a log-normal distribution (with μ=4.29 and σ=1.28) and peer inter-arrival times from a Pareto distribution (with a=2.52 and b=1.55) as reported by recent



(a)

(b)

FIGURE 31.: Effect of churn: (a) and (b) Mean interval between ancestor change as a function of peer population for mesh- and tree-based approach, respectively. Each line is for a different incoming degree.

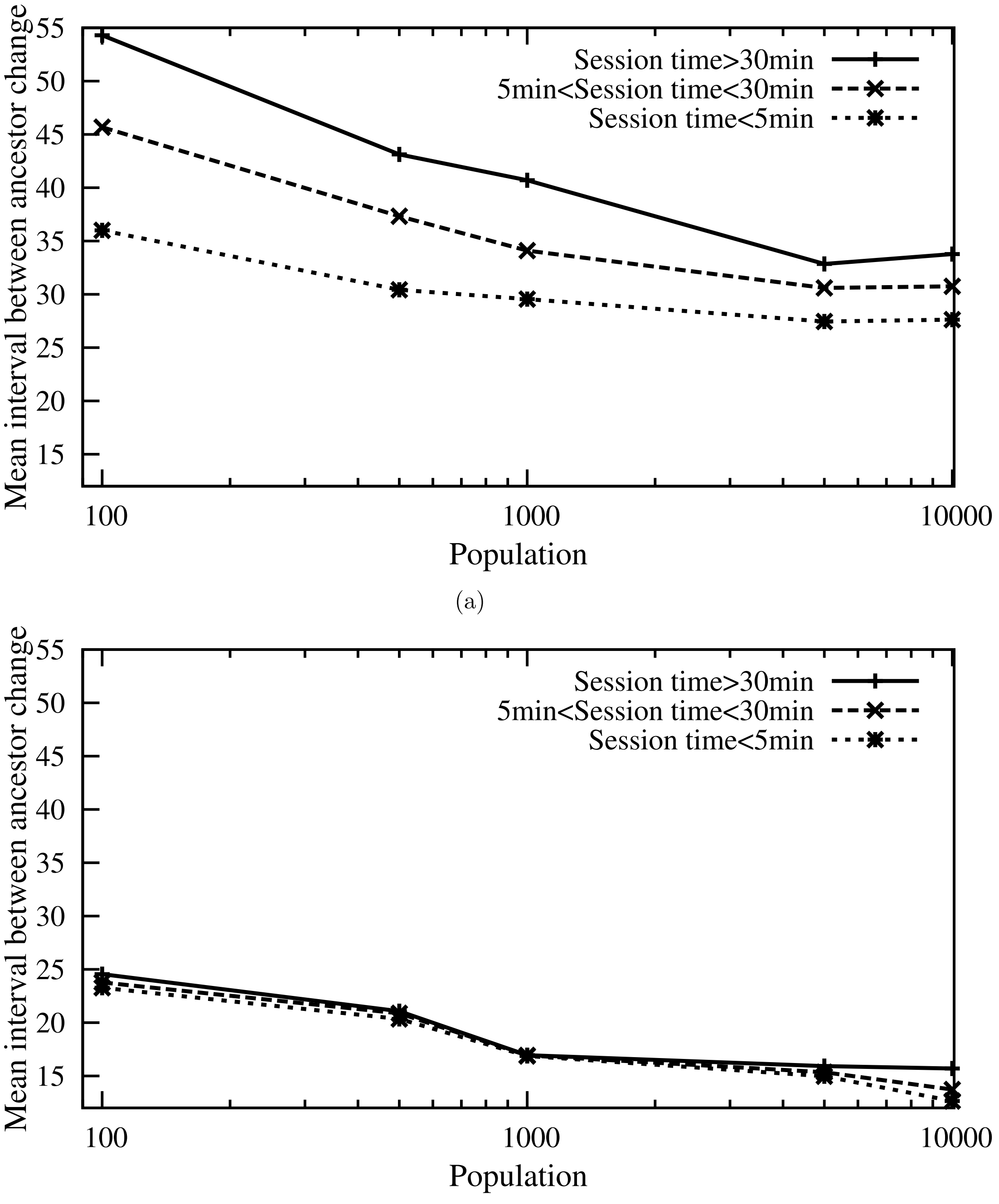
empirical studies [129, 87]. The length of each simulation is 6000 seconds to model a roughly 2-hour event. Presented results are measured at the steady state and averaged over multiple simulations with different random seeds.

4.5.2.1. Ancestor Changing Rate

Figures 31.(a) and 31.(b) depict the mean interval between ancestor changes as a function of peer population in the steady state for three different peer degrees in both mesh- and tree-based approaches, respectively. In the tree-based approach, the ancestor nodes consist of both direct parents as well as any upstream nodes on the path from source. In the mesh-based approach, the ancestor nodes include direct parents as well as any upstream node on the diffusion subtree[1].

These figures demonstrate that the path from source to individual peers is more stable in the mesh-based approach (20%-70%) than in the tree-based approach (5%-40%). The ancestor changing rate increases with the peer degree since the larger number of parents increases the likelihood that one of them leaves the system. Furthermore, for a specific peer degree, the ancestor changing rate increases with peer population. This is mainly due to the fact that the average distance of individual peers increases with peer population in both approaches. Figures 31.(a) and 31.(b) also show that the slope of change in stability is higher for smaller peer degrees due to the stronger effect of population on overlay depth in these scenarios.

[1] Note that the notion of ancestor is not well defined for swarming subtrees since swarming ancestors are dynamically determined. Furthermore, our result in Figure 30. showed that the departure of swarming ancestors does not significantly affect content delivery.



(a)

(b)

FIGURE 32.: Effect of churn: (a) and (b) Mean interval between ancestor change as a function of peer population for mesh- and tree-based approach, respectively. Each line shows the ancestor change among a specific group of peers based on their session time. Peer incoming degree is 8.

An interesting question is *"whether the observed ancestor changing rate for individual peers is correlated with their session times?"*. To investigate this issue, we divide all peers into three groups based on their session times (st) as follows: *(i)* 30min$<st$, *(ii)* 30min$\leq st \leq$5min, and *(iii)* $st<$5min. Figures 32.(a) and 32.(b) depict mean interval between ancestor change within each one of these three groups for both approaches with peer degree 8.

In the mesh-based approach, peers with higher session times on average experience a higher degree of stability among their ancestor. This is primarily due to the fact that once a connection is established between two long-lived peers, it remains in place for a long period of time. This enables long-lived peers to gradually move to higher levels of the overlay and improves the stability of higher levels. However, in the tree-based approach, there is no visible correlation between the ancestor changing rate and peer session time since all three groups exhibit roughly the same ancestor changing rate across different degrees. This is the direct result of maintaining diverse trees. By forcing each peer to be an internal node in one tree and leaf node in all other trees, the departure of each peer causes instability for all the downstream nodes on the tree where it serves as an internal node.

4.5.2.2. Frequency of Deadlock Event

As we explained in Subsection 4.2.1., a deadlock event occurs in the tree-based approach when a tree becomes saturated and can not accept a newly arriving (or partitioned) leaf peer. Figure 33. shows the average percentage of leaf peers that experienced deadlock as a function of peer population for three different number peer degrees. This figure indicates that the percentage of deadlock events drops as the peer degree decreases or the peer population increases. Increasing peer population increases the number of leaf peers that a tree can accommodate and thus reduces the percentage of deadlock events. Increasing peer degree has an opposite effect since it increases the average number of partitioned leaf peers when an internal node departs. This higher rate of partitioning events among leaf nodes leads to a larger percentage of deadlock events for higher degrees. Figure 33. also shows that in a group of 1000 peers with peer degree 8, on average 40% of join (or rejoin) attempts results in a deadlock. This implies that a peer may remain partitioned from one or more trees for an extended period of time. We further quantify the partitioned intervals in the next subsection.

4.5.2.3. Average Peer Connectivity

None of the above metrics properly capture the duration of partitioning intervals for those peers that may not be able to quickly connect to the desired number of parents due to deadlock events in the tree-based approach or inefficient peer discovery in the mesh-based approach. To properly quantify the degree of connectivity for individual peers, we keep track of the weighted average incoming degree of individual peers over time. Each spike in Figures 34.(a), presents the distribution of weighted

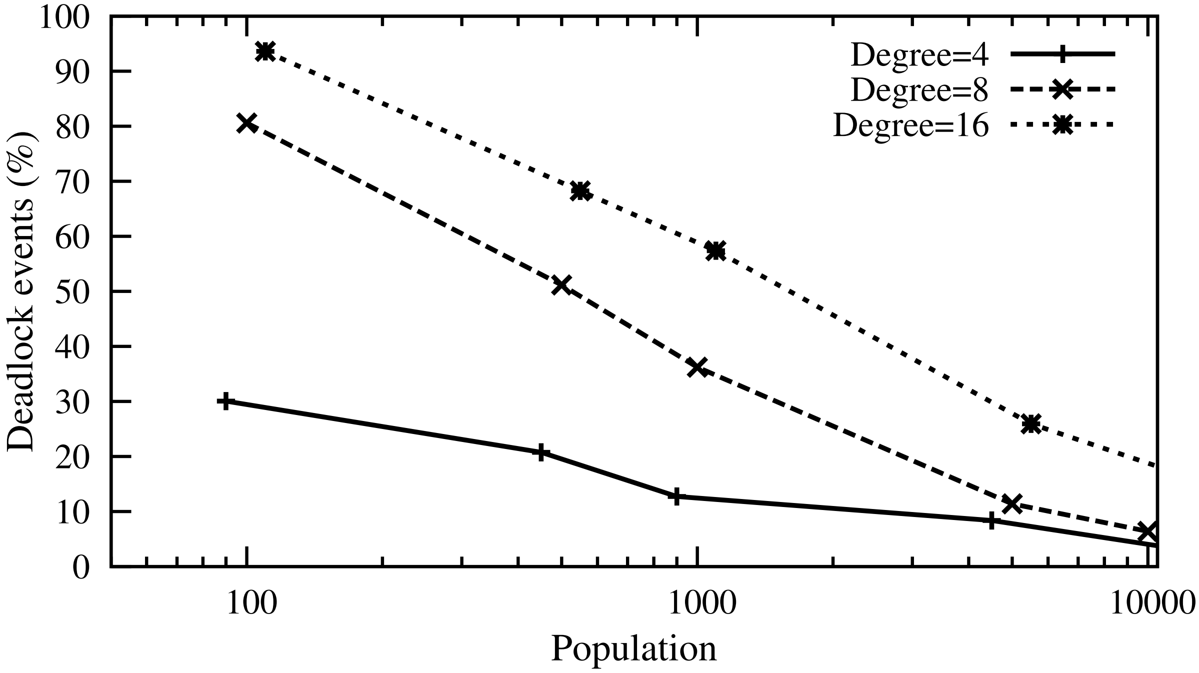


FIGURE 33.: Percentage of deadlock events in the tree-based approach.

average incoming degree for both approaches across a group of 100 peers with a particular target peer degree (4, 8, 16). Figures 34.(b) and 34.(c) presents the same concept for peer population of 1000 and 10,000 peers respectively. These figures reveal that the mesh-based approach enables individual peers to reach much closer to their target degree despite churn. As the peer degree increases, the gap between the average degree and the target peer degree grows in both approaches but due to different reasons.

In the tree-based approach, the percentage of deadlock events increases with the peer degree (as we showed in Figure 33.) which results in extended partitioning intervals for a significant fraction of peers and thus limits their average degree. In

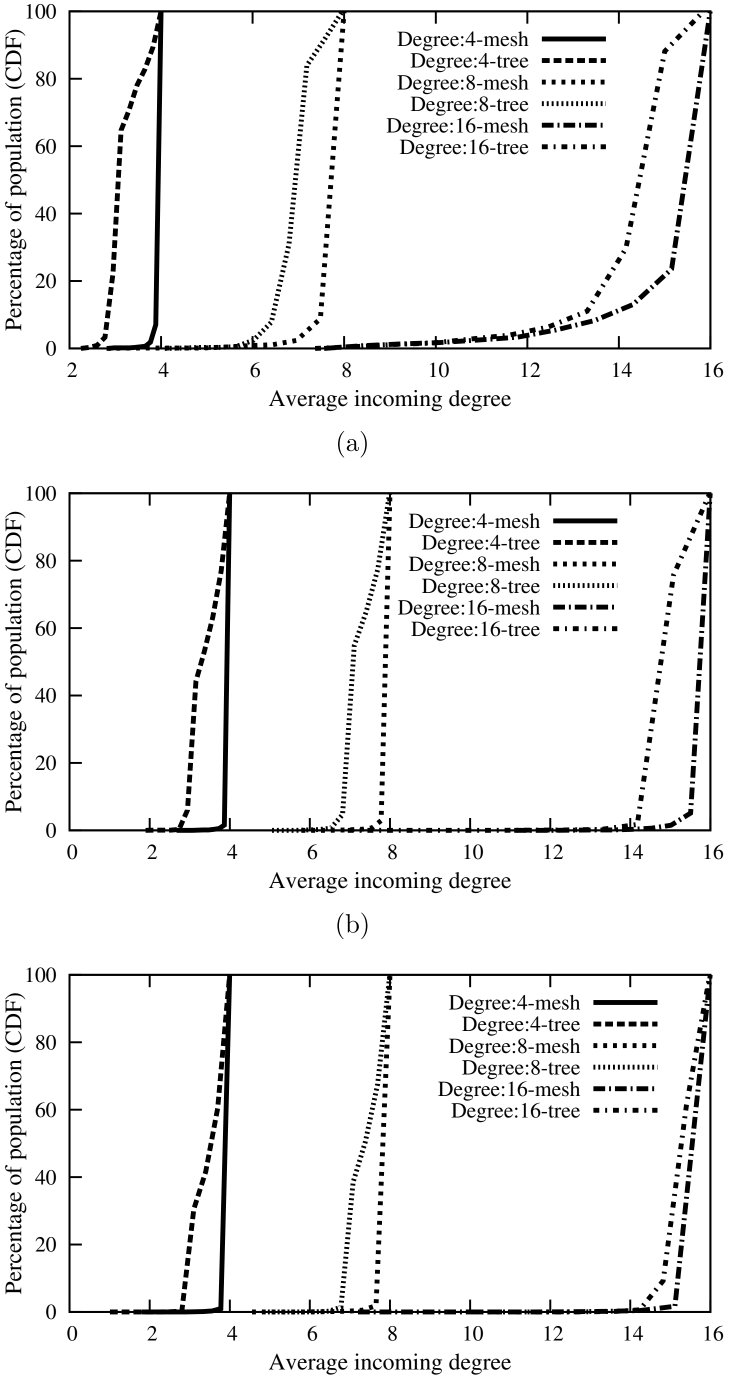


FIGURE 34.: Effect of churn: (a), (b) and (c) CDF of weighted average incoming degree for three degrees of 4, 8 and 16 when peer population is 100, 1000 and 10000, respectively.

the context of mesh-based approach, as the target peer degree increases, it becomes increasingly more difficult for participating peers to maintain their incoming degree at the target level. At any point of time, a significant fraction of peers are in the state of flux, searching for more parents to reach their target degree. The absolute gap between two approaches narrows by increasing peer degree simply because the average degree in the mesh-based approach experiences a larger drop. Interestingly, while the *absolute* gap between the average degree and the target degree widens with peer degree in both approaches, the ratio of the average degree to the target degree which is a better indication for delivered quality, is indeed increasing with the target peer degree. In a nutshell, the delivered quality in both approaches should increase with peer degree.

4.6. Summary

In this chapter, we compared the performance of tree-based and mesh-based P2P streaming approaches through simulations. We illustrates the similarities and differences between these approaches. Further, we evaluated the performance of their content delivery mechanisms over a properly connected overlay. We also investigated their ability to cope with churn, in particular the performance of their content delivery over a distorted overlay and the cohesion of their overlay under persistent churn.

Our evaluations reveal that swarming content delivery in mesh-based approach exhibits a superior performance across a wide range of scenarios. This is primarily due to the ability of the swarming mechanism to minimize the impact of a low bandwidth connection on the connected child peer by providing the required content through other parents. In contrast, the tree-based approach requires each description of the content to be delivered through a particular tree which extends the adverse effect of a low bandwidth connection to all its downstream peers on that tree. Overall, our results show that the tree-based approach is sensitive to the ratio of peer bandwidth to description bandwidth. This implies that the tree-based approach has a sweet spot for peer bandwidth where it can effectively utilize available resources and provide the desired quality. More interestingly, in the mesh-based approach, the longer a peer remains in the system, the higher the degree of stability it experiences, and thus the higher the delivered quality it receives.

Overall, our comparison study reveals that the mesh-based approach is suitable for scalable delivery of live streaming in dynamic and resource constrained networks. On the other hand, tree-based approach can be adopted for static and high provisioned networks which resulted in shorter playback delay and less signaling overhead.

In this chapter, we compared tree and mesh-based approaches, in the next chapter, we focus on the comparison of existing mesh-based P2P streaming mechanisms for delivery of live content.

CHAPTER V

COMPARATIVE STUDY ON MESH-BASED LIVE P2P STREAMING MECHANISMS

Material in this chapter was adapted from a paper intended to be published in a journal. The paper is co-authored with Prof. Reza Rejaie. The experimental work is entirely mine. The text is mostly mine with some contributions from Prof. Reza Rejaie.

There exists many mesh-based P2P streaming solutions for live content with various scheduling schemes that are evaluated under different settings. However, to our knowledge, systematical comparison between these solutions have not been performed yet. In this chapter, our goal is to perform a head-to-head comparison of the design spectrum of block scheduling schemes. Towards that, we introduce a novel evaluation methodology that helps us in doing such a systematic comparison and explaining the underlying root causes for the observed performance.

5.1. Motivation & Contributions

The key component of mesh-based P2P streaming is a *block scheduling* scheme at individual peers that determines which subset of blocks should be pulled from each one of the parents. Designing an effect block scheduling is challenging due to the conflicting requirements of live P2P streaming which are the followings: *(i)* : the in-time delivery of each block to individual peers, and *(ii)*: the diversity of available blocks among peers in order to enable effective swarming. In essence, addressing the timing requirement demands for pulling missing blocks with earlier playout time whereas addressing the diversity requirement demands for pulling missing blocks in a random (or rarest-first) fashion. Another difficulty in the context of live streaming is that the pool of newly available blocks for delivery is very small (compared to the entire content in file swarming). This largely restricts the degree of diversity in available blocks among peers and limits the opportunity for swarming content delivery. A block scheduling scheme for a mesh-based P2P streaming mechanism of live content should be carefully designed to addresses the above challenging requirements. However, the availability of excess resources (*i.e.*, source and peer bandwidth) or/and large buffer at each peer could relax the above requirements by reducing the distance from source and providing more opportunities for delivery of each block.

A few recent studies have proposed new mesh-based P2P streaming mechanisms that incorporates a variety of scheduling schemes ranging from simple pull-rarest-first [64, 66, 71, 70] to prioritizing blocks based on various combinations of their playtime and rarity [68, 67, 130]. These studies often evaluate their proposed mechanisms through simulations in a resource-rich scenario [65] or through actual deployment where available resources (especially peer bandwidth) may not be known. Despite the challenges in accommodating the conflicting requirements for block scheduling, these studies have often reported high delivered quality to participating peers. This raises the following important question: *"Does the reported performance in previous studies on mesh-based P2P streaming mechanisms represent the intrinsic ability of their scheduling scheme to utilize available resources or is it merely the side effect of abundant resources in their evaluation? How does additional resources affect the performance of simple scheduling schemes?"*.

In a nutshell, the following two important issues about proposed mesh-based P2P streaming mechanisms of live content have not been adequately addressed by previous studies:

- *Effect of Block Scheduling Scheme*: How does a block scheduling scheme perform in a scenario with limited resources? What aspects of a block scheduling scheme primarily result in the observed (good or bad) performance?

- *Effect of Available Resources*: How does the availability of various types of excess resources (*i.e.*, peer and source bandwidth) affect the performance of a block scheduling scheme? What are the underlying causes for the observed effect of excess resources?

The contributions of this chapter are as follows: First, it presents an intuitive methodology for evaluation of mesh-based P2P streaming mechanisms that clearly demonstrates the underlying performance bottlenecks. We leverage our insight from PRIME in Chapter III, and propose an evaluation methodology to properly capture the global pattern of content delivery. Further, we derive the proper pattern of content delivery that minimizes the required resources and buffer-size in the system, and present a set of characteristics to identify such a pattern, *i.e.*, the signature of a proper pattern of delivery. The proper pattern of content delivery then serves as a reference to identify the underlying causes in the observed performance of various scheduling schemes.

Second, leveraging our evaluation methodology, we dissect the performance of mesh-based P2P streaming mechanisms of live content to systematically examine the impact of block scheduling scheme and different overlay structures in realistic settings that are both resource-constraint and resource-rich on their overall performance. Towards that, we identify the design space of the block scheduling schemes by exploring different ways to address the conflicting requirement between timing and diversity in timing aware swarming mechanisms. Then, we select several candidate scheduling schemes that represent the entire design space as well as the key features in the previously proposed scheduling schemes. Using our evaluation methodology, we examine the performance of each candidate schemes in both resource constraint and resource-rich environments. This illustrates the ability of our methodology to assess the separate effect of scheduling scheme and available resources on the performance of mesh-based P2P streaming mechanisms.

5.2. Block Scheduling: Design Space

In this section, we identify the design space for block scheduling schemes in mesh-based P2P streaming of live content, and then select a few candidate schemes that properly represent interesting scheduling schemes across the space. The block scheduling at each peer should determine the requested blocks from each parent based on the following information:

- the missing blocks that still have sufficient time to be pulled,
- available blocks among parents based on their reports, and
- congestion controlled bandwidth from each parent that is passively measured by each child.

The total number of requested blocks from all parents during one interval (*i.e.*, block budget) is determined by the aggregate bandwidth from all parents, while the number of pulled blocks from each parent is proportional to its contributed bandwidth.

The goal of the scheduling scheme at each peer is to ensure in-time delivery of requested blocks while addressing the diversity of available blocks among peers. The block scheduling function is invoked once per Δ seconds and in each scheduling

Parameter	Description
t_p	Peer's playout time.
t_{src}	Source's playout time.
t_{last}	Largest available timestamp in the peer.
t_{new}	Largest reported timestamp by parents.
ω	Available buffer at each peer.
Δ	Period of scheduling events.

TABLE 7.: Summary of used parameters.

event it considers blocks within its current window of ω seconds (buffer) that should be pulled from parents in the current interval. The timestamp of the blocks in the current buffer falls within the following range [t_p+Δ, t_p+Δ+ω]. [1] Figure 35. depicts a view of blocks with relevant timestamps (buffer state) for a peer at an scheduling event. Table 7. summarizes the parameters and their descriptions used through this chapter.

Careful examination of Figure 35. reveals that the buffer consists of three distinct regions (*i.e.*, range of timestamps) as follows:

- *Playing Region*: this is the left most region with timestamps within the following range [t_p+Δ, t_p+$2 * \Delta$]. We call this playing region since all these blocks are being played in the next interval. Therefore, explicitly requesting any missing block from this region explicitly addresses the timing requirement.

- *New Region*: this is the right most region with timestamps within the following range [t_{last}, t_{new}] which represents all the new blocks with largest timestamps

[1] Blocks with timestamp [t_p,t_p+Δ] are being played during this interval.

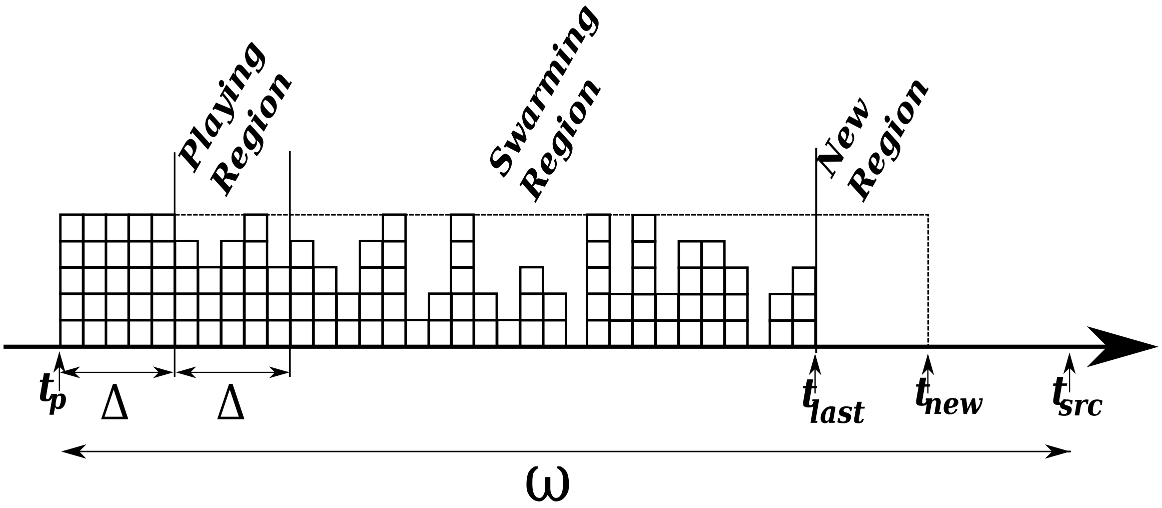


FIGURE 35.: Buffer state at a scheduling event in a peer.

that have become available among parents since the last scheduling event. Requesting these blocks explicitly expands the pool of new blocks which in turn facilitates the diversity of delivered blocks to individual peers.

- *Swarming Region*: This is a larger region in the middle. Since there is no preference among missing blocks in this region, these missing blocks can be requested in a random/rarest-first fashion in order to address diversity. The relatively large size of swarming region provides opportunity to diversify available blocks in this region among peers.

Given the above properties of blocks in these three regions, the design of a block scheduling scheme has the following two dimensions: *(i)* the relative priority (*i.e.*, the order of requesting missing blocks) of different regions, and *(ii)* the choice of *random* or *rarest-first* strategy to select a subset of blocks that are missing but available among parents. Note that setting relative priorities for different regions implicitly controls the allocated block budget to each region.

5.3. Block Scheduling: Candidate Schemes

The mentioned two dimensions of design space for block scheduling scheme, motivate the following eight *candidate* scheduling schemes:

- *Rare* or *Rand*: These schemes select all the blocks from the entire window using a rarest-first (*e.g.*, Coolstreaming [64] or PULSE [66]) or random (*e.g.*, BitTorrent or Chainsaw [65]) strategy, respectively. By enforcing random/rarest-first strategy across the entire window, these schemes maximize the diversity among delivered blocks to different peers. These schemes implicitly address in-time delivery (or timing) of blocks since the number of opportunities to request a block is equal to the number of scheduling events that its timestamp has appeared within the window. This number is larger for blocks with earlier timestamp and thus they are more likely to have been requested.

- *PRare* or *PRand*: These schemes explicitly address the timing requirement (similar to [67]) by first requesting all the missing blocks in the playing region, and then using the remaining block budget to select rare/random blocks from the rest of the window $[t_p+2*\Delta, t_{new}]$

- *NRare* or *NRand*: These schemes explicitly target ensuring the availability of new blocks by first requesting all the new blocks (from the new region), and then using the remaining budget to request a rare/random subset of blocks from the rest of the window $[t_p+\Delta, t_{last}]$.

- *NPRare* or *NPRand*: These are hybrid schemes [4] that first request all the available blocks from the new region, then all the missing blocks from the playing region, and finally use any remaining budget to request a rare/random subset of blocks from the swarming region. Therefore, these schemes explicitly address both timing and availability[2].

The output of each block scheduling scheme is an ordered list of required blocks that are available among parents and should be requested. The next step is *parent selection* where selected blocks are mapped to request from individual parents. Toward this end, we assign each block to a parent that can provide the block, and a smaller fraction of its block budget has been assigned so far. This assignment policy tends to balance the number of assigned blocks to individual parents proportional to their block budgets and exhibits the best performance compared to other policies as we illustrated in Chapter III.

[2] The other possible hybrid scheduling schemes that give higher priority to playing region, namely *PNRand* and *PNRare*, have a performance similar to *PRare* and *PRAND*, and therefore are not considered.

Clearly, one can design other scheduling schemes that balances the conflict between timing and diversity differently [68]. However, we believe that our candidate schemes allow us to properly explore the importance of addressing the timing and diversity requirement in an implicit or explicit fashion, and thus identify fundamental design tradeoffs. Furthermore, our candidate schemes adequately resemble most of the commonly used scheduling schemes in previous studies.

5.4. Evaluation Methodology

In mesh-based P2P streaming systems, it is generally difficult to determine the main underlying causes that limits the performance. To reliably assess the performance of a mesh-based live P2P streaming system, we need an evaluation methodology that properly dissects the inherent abilities of a scheduling scheme from the improvement caused by excess resources. Our earlier discussion on design and evaluation of a new mesh-based P2P streaming mechanism in Chapter III has inspired the following observation: the evaluation methodology for mesh-based P2P streaming mechanisms should capture the global pattern of content delivery since this pattern directly determines timing, availability and diversity of delivered blocks to participating peers. This in turn provides a useful insight to identify the underlying causes for the observed performance by a block scheduling scheme.

In this section, first we present a proper view and a set of metrics to capture the global pattern of content delivery. Second, we present the appropriate pattern

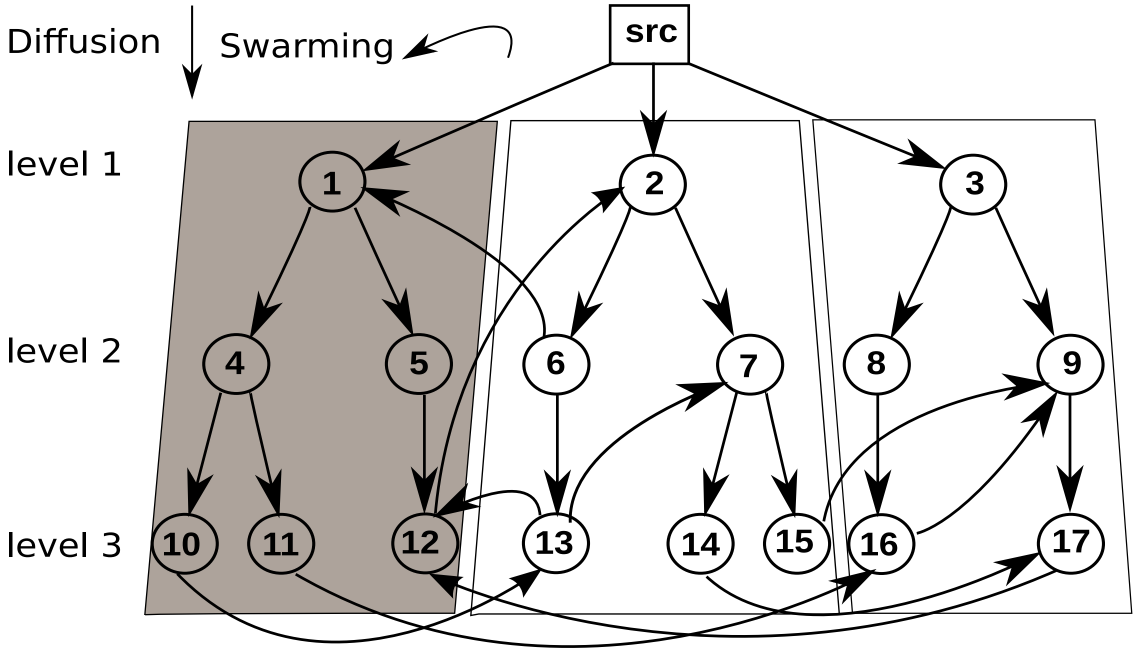


FIGURE 36.: Organized view of a random mesh overlay.

of content delivery that maximizes the utilization of resources and derives the key characteristics or *signature* of such a pattern. We then sketch our methodology for evaluating different block scheduling schemes and present our simulation settings.

5.4.1. Proper View

We leverage the "organized view" of a randomly connected mesh as discussed in Subsection 3.4.2. to properly observe the global pattern of content delivery throughout the overlay. Recall that, in the organized view, participating peers are grouped into *levels* based on their shortest distance (in hops) from source through the overlay as shown in Figure 36.. Peers in level 1 are directly connected to source, peers in level 2 are two hops away from source and so on. Any newly generated block at source must be delivered to levels of the overlay in a sequential fashion, *i.e.*, pulled by different peers at level 1, then by different peers at level 2 and so on. In essence, the organized view clearly illustrates the direction that newly generated blocks should flow through different levels away from source.

Since blocks are pulled along the direction of arrow for each connection, the organized view of an overlay clearly illustrates the direction that newly generated blocks should flow through different levels to reach all peers. In order to effectively utilize outgoing bandwidth of all peers, each peer should always have sufficient amount of useful content for delivery to its children. This observation motivates diffusion and swarming phases for pattern of content delivery through the organized overlay that has been discussed in Subsection 3.4.3.. Recall that, in the *diffusion phase*, the content is pulled to lower levels away from the source. At the end of a diffusion phase for a segment, a subset of blocks for a given segment are available at all peers from the corresponding diffusion subtrees. During the *swarming phase* each peer pulls content of other diffusion subtrees from its swarming parents. At the global level, content of a given diffusion subtree is pulled by peers in other diffusion subtrees during the swarming phase. Therefore, the total number of intervals for delivery of blocks to each peer is equal to the number of diffusion intervals plus the required number of swarming intervals.

5.4.2. Performance Metrics

To capture the global pattern of content delivery, we introduce a set of metrics that are leveraged by the intuition of two-phase content delivery over the organized view of the overlay.

We use two metrics to capture the behavior of diffusion phase and a single metric to capture the behavior of swarming phase as follows:

- *Diffusion rate* of level i presents the rate of arrival of *new* blocks to peers in level i. To capture the diffusion rate of a level, we only capture the first copy of each block that arrives at that level.

- *Diffusion time* of a block to level i is the time that elapses from its generation time at source until the first copy of this block is arrived to a peer in level i. We present the diffusion time in terms of the number of intervals (Δ) to provide an easy comparison with periodic pulling of blocks by individual peers.

- *Block Availability* for peer p represents all the available blocks among p's parents. We show the availability across the buffer state (Figure 35.) as a percentage of all blocks in each window that are available among parents. To derive block availability among peers in level i, we average the availability of blocks across the peers in level i for each window of the buffer state. Therefore, block availability of level i presents an average notion of content availability among parents of peers in that level.

5.4.3. Signature of a Good Pattern of Delivery

We derive the signature of patterns of content delivery that leads to a good performance by deriving the value of per-level diffusion rate, the distribution of per-level diffusion time, and the block availability for such a pattern. The three signatures of a good diffusion phase is as follows:

$DiffusionRate(i)$ = *Stream Rate*

The diffusion rate to *all* levels of the overlay must be equal (or very close) to stream bandwidth. This condition ensures that peers in all levels continuously receive new blocks. The continuous availability of new blocks ensures the diversity of available blocks for effective swarming as well.

$DiffusionTime(i) \leq i$

Given the mis-alignment of block generation time and pulling intervals, a new block can be pulled to level 1 within one interval Δ after each generation time. Delivery of each block to each lower level requires at least an additional interval. Therefore, in general diffusion time of a block to level i ($DiffusionTime(i)$) should be less than i intervals. Therefore, the minimum required intervals for diffusion is *depth* intervals.

Diagonal Block Availability Across windows

Deriving a signature for proper content availability during the swarming phase is more subtle and requires a deeper understanding of content delivery. With a well-designed block scheduling scheme there are two observations regarding the length of the swarming phase and visibility horizon of peers: First, If a peer has $deg - 1$ swarming parents in different diffusion subtrees, it can receive all the blocks of a segment in one interval. In a randomly connected overlay, some swarming parents of a peer may reside in the same diffusion subtrees. Using both modeling [64] and simulations in Chapter III, it has been shown that the required number of swarming intervals in a randomly connected overlay is 3. Second, the range of available blocks to peers in level 1 (visibility horizon) is exactly $depth$+3 intervals. But it decreases by one interval as we go to lower level. In general case, the range of available blocks to peers in level i is equal to ($depth$+4-i) intervals after the playout time. For example, peers in level 1 can see up to window 7 in source (ω is 7 in this example), the view of level 2 peers is up to window 6 and so on. In particular, peers at the bottom level only observe available blocks in the first three intervals (from their swarming parents) and one more interval from their diffusion parent.

The notion of the range of visible blocks for peers at each level and the required length of swarming phase (*i.e.*, 3) enable us to determine content availability to each peer based on the location of its parents. Figure 37.(a) (the solid lines) shows the percentage of available blocks in each window of buffer for peers in the bottom 3 levels of the overlay with $depth$ of 4. Typically, each peer in level i has one diffusion parent in level $i-1$ and $deg-1$ swarming parents in the bottom level. Therefore, the available content to a peer in the bottom level (L_4 in Figure 37.(a)) is limited to the first four windows. The availability during the first three windows is linearly decreasing since blocks in earlier windows have had more time to be requested. This represents the

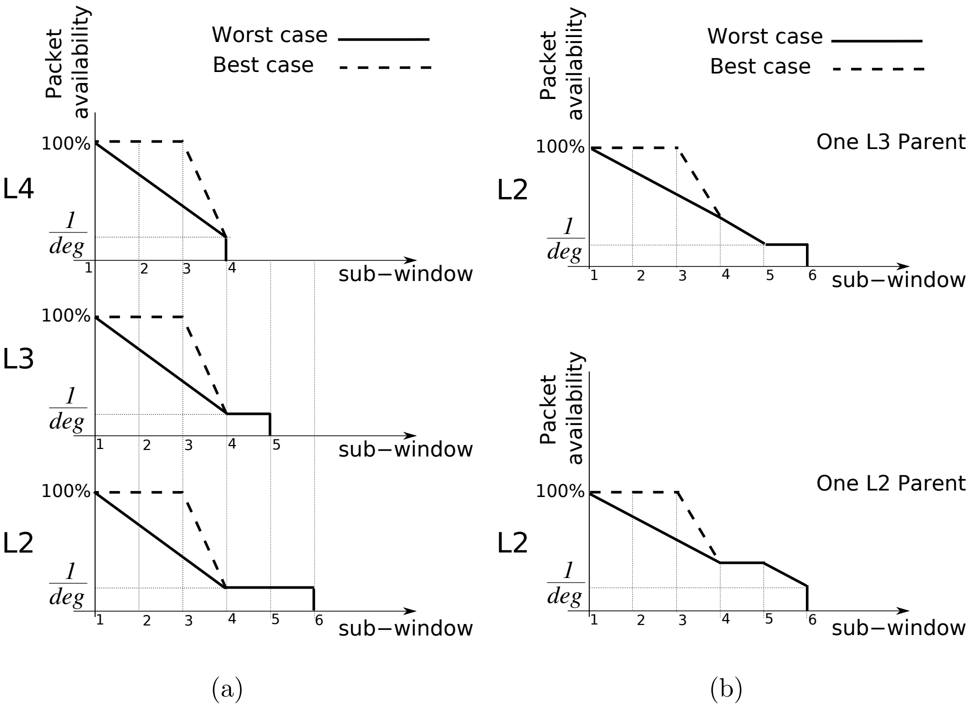


FIGURE 37.: Block availability: (a) Block availability across the bottom 3 levels of the overlay for the best and worst case scenarios. (b) Block availability in level 2 peers, while the top figure is the case with one level 3 parent and the bottom figure is the case with one level 2 parent.

worst case scenario where swarming parents are not in different diffusion subtrees. If all swarming parents are in different diffusion subtrees, then the swarming content can be pulled in one interval. This represents the best case scenario where content availability during the first three windows is 100% as shown with dashed line in Figure 37.(a). The region between the worst and best case scenarios shows an area of possible content availability for good swarming phase among peers in the bottom level. The availability during the fourth window in equal to $(1/deg)$% of blocks that is provided by the diffusion parent.

We can extend the above discussion to derive the signature of content availability among peers in higher levels. For example, for peers in L_3 and L_2, the signature of content availability during the first three windows is similar. The only difference is the higher visibility of content for peers in higher levels. The availability of content during these extra sun-windows is $(1/deg)\%$ of content from their diffusion parent.

In practice, a peer may have swarming parents at other levels than level *depth* (*e.g.*, a peer in level 3 may have a swarming parent in the same level). In these cases, these parents have larger content visibility and thus improve the content availability of their child peer especially in fourth or higher windows. For example, as shown in Figure 37.(b) (the top portion), a peer in level 2 that has one swarming parent in level 3 (as opposed to all swarming parents in level 4), experiences $(1/deg)\%$ higher content availability in the fourth window compared to the case with all swarming parents in level 4 (Figure 37.(a)). However, the same peer with a single swarming parent in level 2 (the bottom portion of Figure 37.(b)) experiences $(1/deg)\%$ higher content availability in both fourth and fifth windows. Note that as the content availability in window k increases by a swarming parent in higher level, this improves the minimum content availability in earlier windows ($<k$) by the same amount.

In summary, the above discussion provides the minimum requirement for content availability for peers in each level in order to have a good-performing swarming phase (shown in Figure 37.(a)). However, any content availability above this line is also *possible due to the location of swarming parents in different diffusion subtrees and in different levels of the overlay.*

5.4.4. Our Methodology

In the next section, we demonstrate the effectiveness of our proposed metrics and signatures in evaluating the performance of mesh-based P2P streaming mechanisms and more importantly revealing the underlying performance bottlenecks. Our evaluation methodology incorporates the following ideas to separate the effect of block scheduling scheme from available resources on system performance. Initially, we focus on a "resource constraint" scenario with minimum source and peer bandwidth in static environments. We examine both homogeneous and heterogeneous scenarios in this setting. Centering on static environments, lets us avoid any potential side effect that churn may have on our findings and represent the best possible performance of the candidate scheduling scheme. We examine all the scheduling schemes over the same randomly connected and directed overlay with the proportional incoming and outgoing degree for all peers. The resource constraint scenario stress-tests a block scheduling scheme and exposes its inherent ability to operate without any excess resources in the system. Further, we illustrate how adding different types of excess resources (*i.e.*, source and peer bandwidth) can improve the performance of those schemes that performed poorly with limited resources. In the next step, we examine the effect of peer dynamics on the performance of various scheduling schemes to illustrate a more realistic scenario. We further examine various overlay structures, namely random or localized.

5.4.5. Simulation Setup

To illustrate the proposed evaluation methodology, we investigate the effect of candidate scheduling schemes and available resources using *ns* simulations. Using packet level simulation is a proper choice because it incorporates packet level dynamics, delay and loss that are essential for meaningful evaluation of swarming content delivery. It also enables us to directly control the available resources in the overlay, and thus reliably derive our conclusions.

In our simulations, physical topology is generated by Brite [126] with 15 AS and 10 routers per AS in top-down mode and RED queue management in all routers. Except when noted, we use the following default settings: Source bandwidth is set to 1 Mbps which is the minimum value for delivery of the stream quality to peers in level 1. The stream is MDC encoded and has 6 descriptions with the same constant bit rate of 160 Kbps. All overlay connections are congestion controlled using RAP [124]. Overlay is uni-directional, consists of 500 peers with symmetric accesslink bandwidth of 960 Kbps and peer degree of 6.

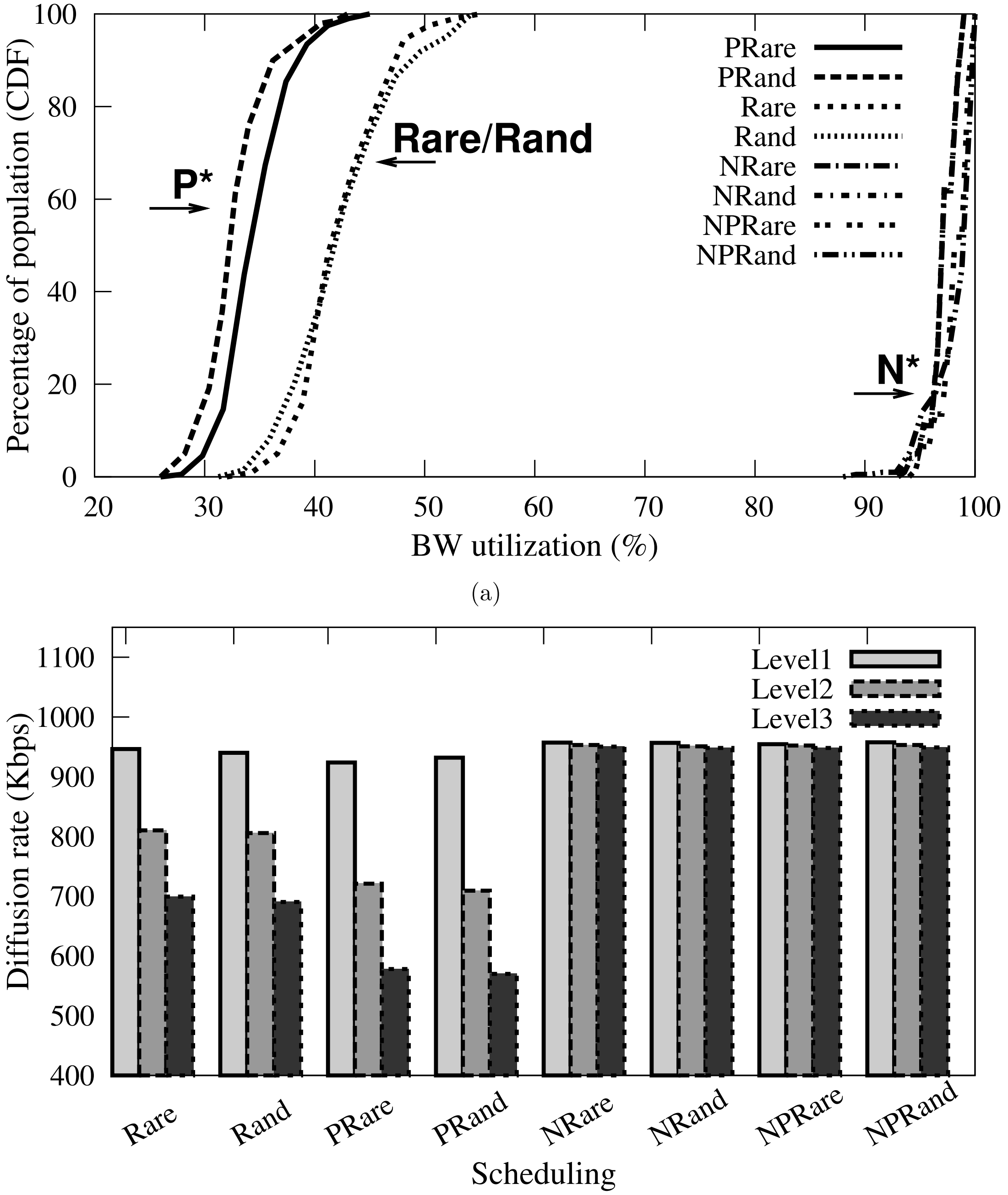
All peers maintain a synchronized playout time which is ω*Δ seconds behind the source's playout time. The value of ω is selected to be the minimum value that can accommodate in-time delivery of blocks for a given population and peer degree (*i.e.*, when the overlay has *depth* levels, the amount of buffering at each peer (ω) is set to its minimum value of $depth + 3$). Furthermore, delay on each access link is randomly selected between $[5ms, 25ms]$ while core links have high bandwidth in the range of 4 to 10 Gbps. This ensures that in our simulations bandwidth bottleneck is always at the edge, and avoids any subtle effect of major congestion in the core. Each simulation is run for 400 seconds.

5.5. Performance Evaluation: Scheduling Schemes

In this section, we examine the performance of our candidate scheduling schemes in the "resource-constraint" settings over a static and then dynamic overlay.

5.5.1. Reference Scenario: Homogeneous Peer Bandwidth

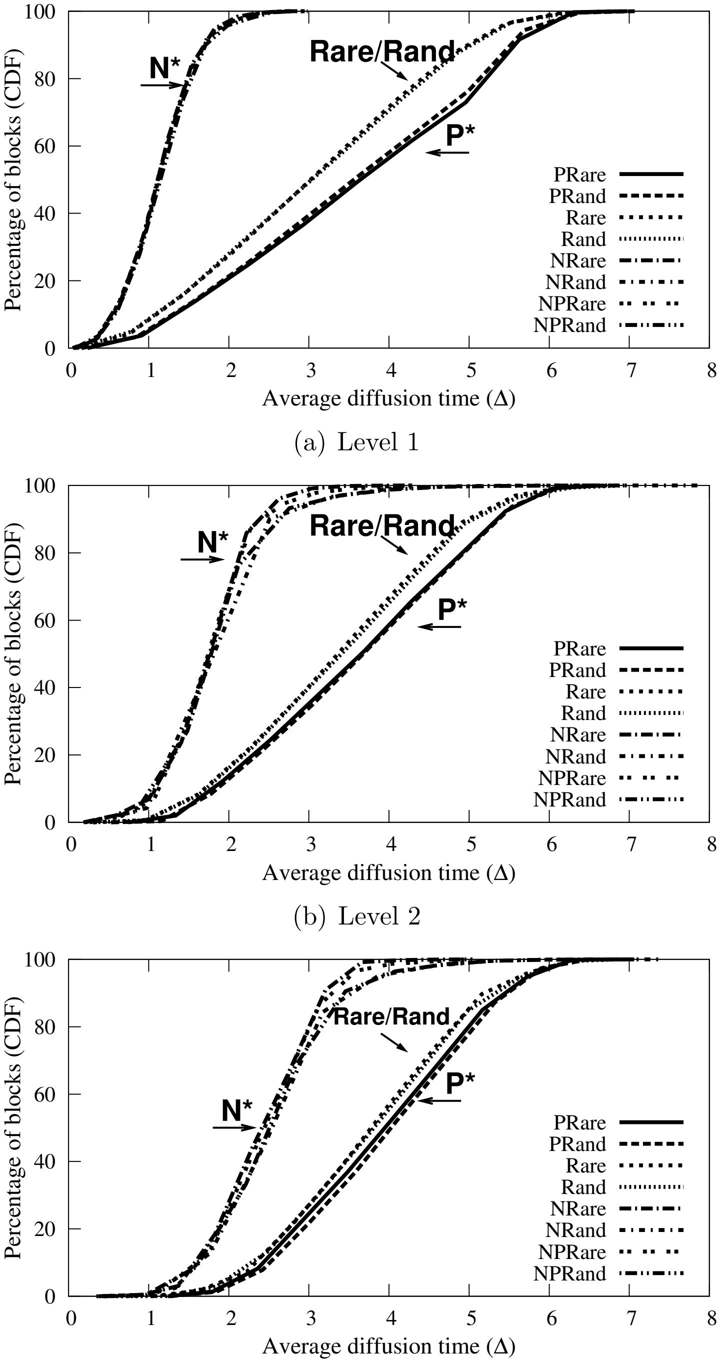
Figure 38.(a) shows the distribution of the incoming bandwidth utilization among participating peers for various scheduling schemes. This figure reveals that mean bandwidth utilization with N* schemes is more than 94% but it drops down to 45% and 35% for Rare/Rand and P* schemes, respectively. To identify the underlying causes for this gap in observed performance, Figure 38.(b) shows the diffusion rate to the top three levels of the overlay for all the eight candidate scheduling schemes. This figure illustrates that all variants of N* schemes achieve high diffusion rate across all



(a)

(b)

FIGURE 38.: Effect of scheduling schemes: (a) Distribution of incoming bandwidth utilization. (b) Average diffusion rate across different levels of the overlay.



(a) Level 1

(b) Level 2

(c) Level 3

FIGURE 39.: Effect of scheduling schemes: Distribution of diffusion time normalized by Δ for different scheduling schemes, (a), (b) and (c) are for level 1, 2 and 3 peers, respectively.

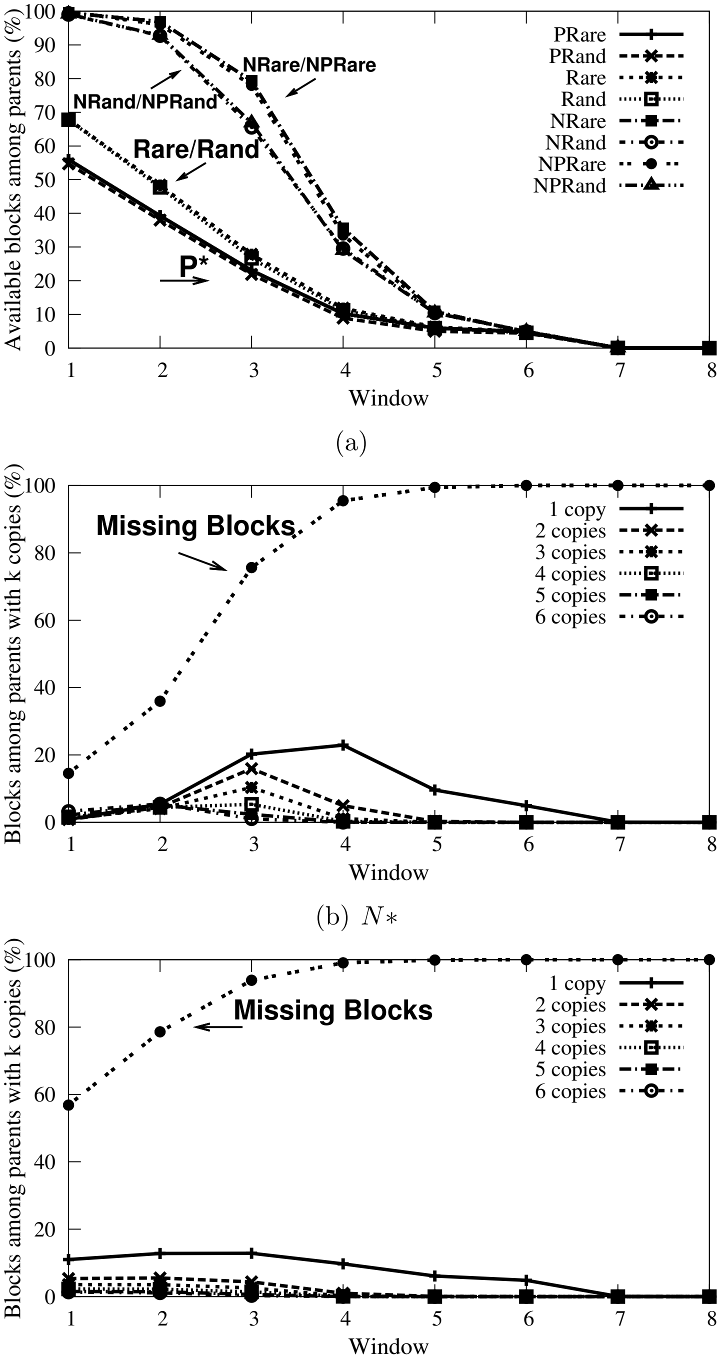
levels of the overlay. However, in other four scheduling schemes, the diffusion rate drops as we go to lower levels of the overlay.

Figures 39.(a), 39.(b) and 39.(c) present the distribution of diffusion time across all delivered blocks to the top three levels (in terms of the number of intervals Δ) and shed a useful light on the pattern of content delivery Interestingly, the diffusion time at these top levels show that all the *N** schemes manage to diffuse the majority of blocks to level l within $l + 1$ intervals. Note that this is the fastest diffusion time for blocks to each level since a block that is generated by source at the beginning of one interval can be delivered to level 1 no later than the end of the next interval, and delivery to lower levels simply adds an additional interval to the diffusion time. The minimum diffusion time for delivery of new blocks to the bottom level in *N** schemes leaves a larger gap before their playout time which allows a larger window for the large population of peers at the bottom level to effectively swam these blocks. This in turn leads to a higher utilization of incoming bandwidth among all peers throughout the overlay as shown in Figure 38.(a).

The diffusion time for both *Rare* and *Rand* schemes that purely swarm the blocks (in Figures 39.(a) and 39.(b)), exhibits a uniform distribution across the entire window (all seven intervals), and does not significantly change across levels. This indicates that in *Rare* and *Rand* schemes new blocks arrive at each level in a random order. While all blocks arrive at level 1 within 7 intervals (or ω), 10% of blocks that arrive during the last interval are late and can not be requested by peers in the lower level. This in turn reduces the diffusion rate to lower levels by an extra 10% as shown in Figure 38.(b). In summary, the late arrival of new blocks to the top level has a propagating effect on the diffusion rate of other levels. Moreover, the diffusion time for only half of the delivered blocks to the bottom level is sufficiently short to swarm. Considering that 80% of blocks are delivered in the lowest level, and only 50% of them can be swarmed within the given buffer size, we have a moderate utilization of incoming bandwidth ($>$ 40%) for 90% of participating peers in *Rare* and *Rand* schemes, which is aligned by the result shown in Figure 38.(a).

The *P** schemes perform slightly worse than *Rare*/*Rand* schemes. As shown in Figure 39.(a), in *P** schemes roughly 20% of blocks arrive at the top-level after six intervals which in turn reduces the diffusion rate to the lower level and results in lower utilization of bandwidth among all peers. Closer examination of *P** schemes revealed that peers in the top level pull a fraction of their required blocks in the playing region from source. These blocks are pulled from source around six intervals after their generation time and thus can not be delivered to lower levels, resulting in drop in the diffusion rate of lower levels.

Content Availability for Individual Peers- We now examine the availability and diversity of content to individual peers in order to better understand the dynamics of content delivery for different scheduling schemes. Figure 40.(a) shows the average percentage of all blocks within each window (Δ) of the buffer that are available among parents of individual peers for all candidate scheduling schemes. This figure



(a)

(b) $N*$

(c) $Rare/Rand$

FIGURE 40.: Effect of scheduling schemes: (a) Available blocks among parents across different windows. (b) and (c) Percentage of number of copies of available blocks among parents that is required for $N*$ and $Rare/Rand$ schemes, respectively.

basically represents the "average view" of available blocks to each peer across different range of timestamps. The figure reveals that *N** schemes achieve a significantly higher degree of content availability among peers especially in windows with lower timestamp. *Rare/Rand* schemes perform slightly better than *P** schemes. Note that rarest-first selection of blocks marginally achieves better availability than random selection by diversifying the blocks in each neighborhood. These figures confirms that poor performance by *P** and *Rare/Rand* schemes is due to the limited content availability among parents of each peer (*i.e.*, content bottleneck).

We now examine the diversity of required blocks among parents of each peer in $N*$ and *Rare/Rand* schemes. Figure 40.(b) and 40.(c) plot the average percentage of blocks with k copies among parents in each window for *N** and *Rare/Rand* scheduling schemes, respectively. Note that the value of k is limited by the incoming degree of each peer. We also show the average percentage of blocks missing by a peer within each window which roughly presents the probability of requesting a block from a given window. These two figures illustrate two important points: First, in both $N*$ and *Rare/Rand* schemes, a significant portion of available blocks are unique (*i.e.*, have a single copy). Therefore, random block selection is likely to select unique blocks. This explains the similarity in the performance of schemes that selects blocks by rarest-first or random strategy. Second, comparing these two figures also reveals that prioritizing the new blocks not only increases the overall availability of blocks but also results in

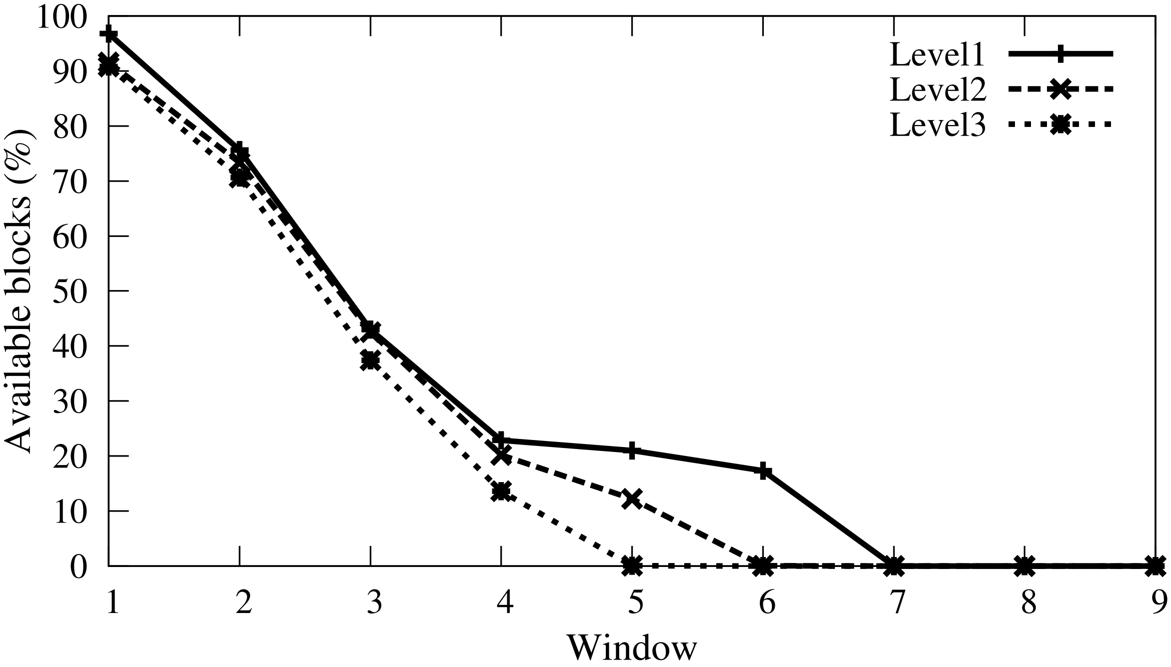


FIGURE 41.: Available blocks in peers from various levels for $N*$ schemes.

a higher diversity of available blocks to individual peers since most of the available blocks to each peer are unique.

Local Pattern of Delivery- Another interesting issue is examining the arrival pattern of the required block in a window at each peer. Towards that Figure 41. shows the average percentage of delivered blocks in each window of the buffer among all peers in a particular level of the overlay in $N*$ scheduling schemes. Since the window slides by Δ seconds (a window) once every Δ seconds, the difference between windows i and $i + 1$ demonstrates the progress in download during window $i + 1$. Figure 41. shows that the rate of progress for different levels are slightly different. During the first *depth* (three in this example) intervals (or windows), peers in each

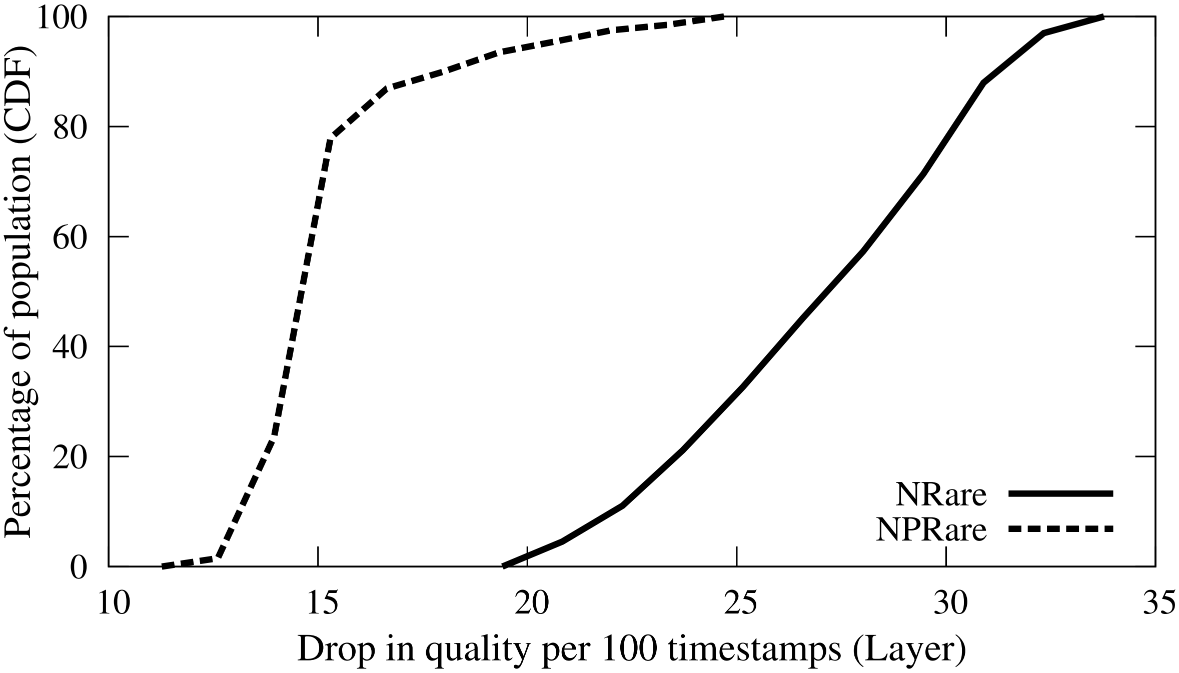


FIGURE 42.: Distribution of variation in quality for $N*$ schemes.

level sequentially receive a fraction (namely $\frac{1}{deg}$% or 16% in this example) of the blocks within their last window. This corresponds to the diffusion rate to each level. During the last ω-*depth* windows (three in this example), all peers experience a rapid rate of progress and receive an equal fraction of remaining blocks in each window. During these windows some blocks are available at each peer and swarming occurs.

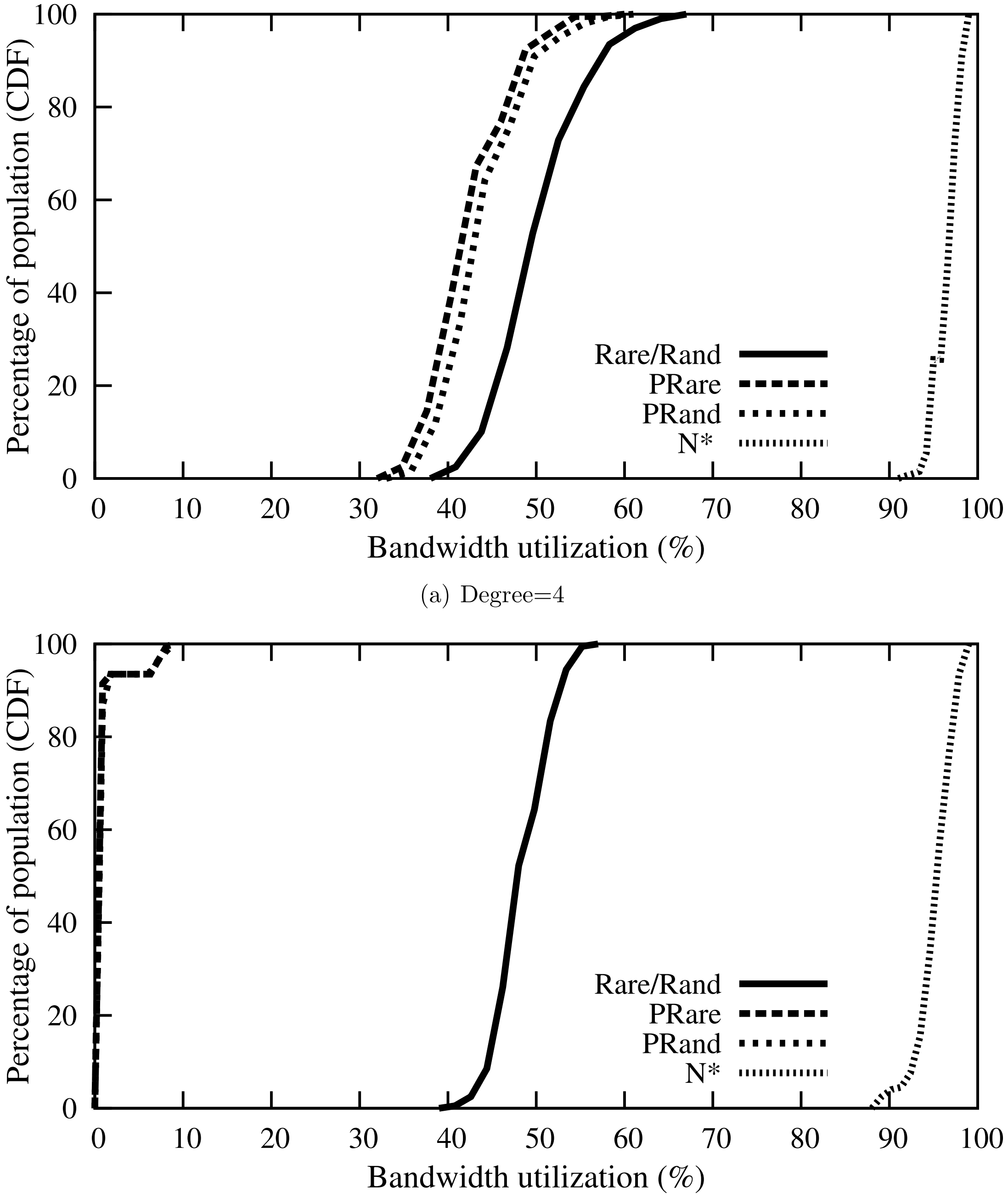
The above findings collectively illustrate that while the $N*$ schemes achieve high diffusion rate through all levels, most of the content delivery actually occurs during the swarming intervals.

Importance of Explicit Timing- Our results have indicated that all $N*$ scheduling schemes that prioritize new blocks similarly exhibit good performance. This raises the question "*whether explicitly requesting the required blocks in the playing region has any effect on the performance of $N*$ schemes?*", *i.e.*, is there any difference between *N** and *NP** schemes? Figure 42. depicts the distribution of variations in delivered quality in terms of average changes in the number of delivered descriptions per 100 blocks among all peers in *NPRare* and *NRare* schemes. This figure shows that the percentage of missing blocks for those schemes that explicitly request playing blocks ($NP*$) is significantly lower than those implicitly address timing requirements. This difference is due to the fact that despite multiple opportunities for requesting each block in *N** schemes, there is still some chances that a portion of the required blocks are not requested. The *P** schemes that explicitly request the missing blocks in the playing region can fill these holes and ensure the stability of delivered quality.

5.5.2. Peer Degree

Next, our goal is to examining the performance of our candidate scheduling schemes in our default "resource-constraint" scenario with peer degree of 4 and 10.

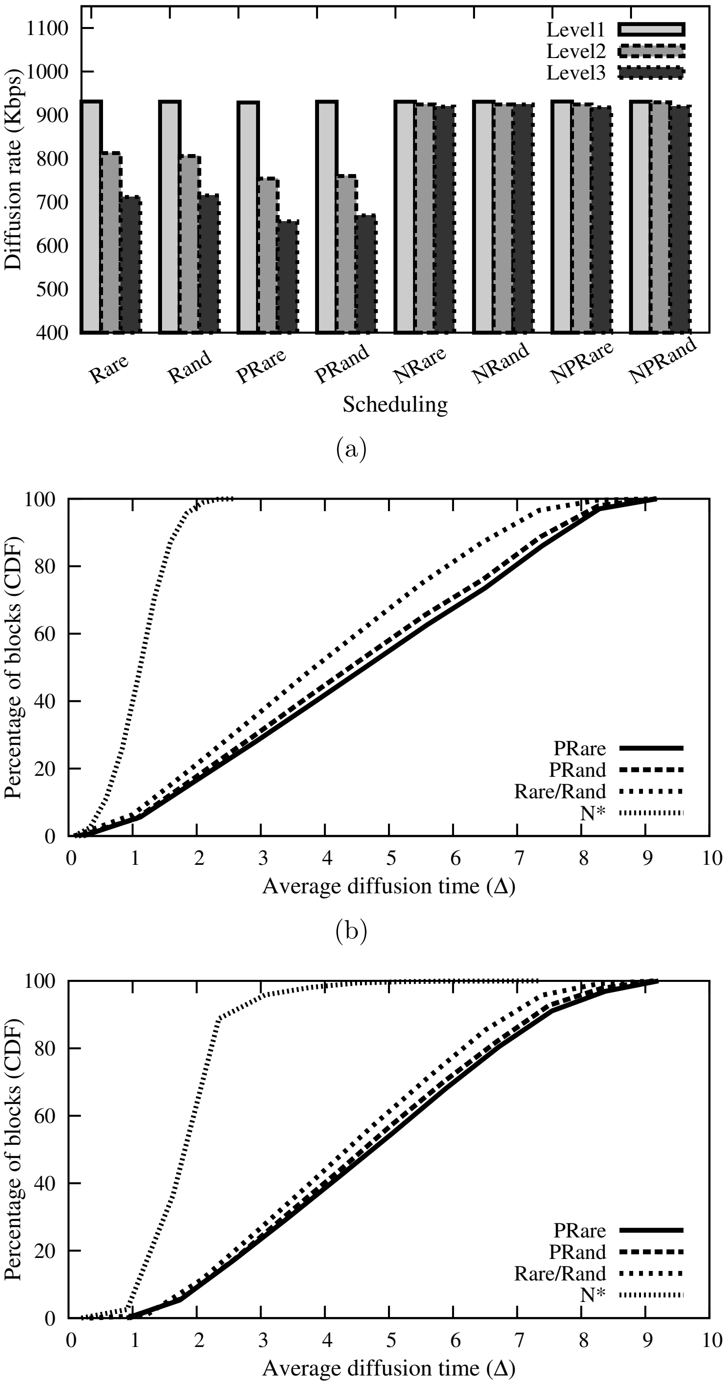
Figures 43.(a) and 43.(b) depict the distribution of bandwidth utilization for various scheduling scheme with peer degree of 4 and 10, respectively. From these figures, we can see that the trend in the performance of various schedulings is independent of peer degree. However, *PRare* and *PRand* schemes perform worse by increase in the peer degree.



(a) Degree=4

(b) Degree=10

FIGURE 43.: Effect of peer degree: (a) and (b) Distribution of utilization of bandwidth across various scheduling schemes for peer degree 4 and 10, respectively.



(a)

(b)

(c)

FIGURE 44.: Effect of peer degree: (a) Diffusion rate across different levels for various schedulings with peer degree 4. (b) and (c) Distribution of diffusion time for various schedulings when peer degree is 4 across level 1 and 2, respectively.

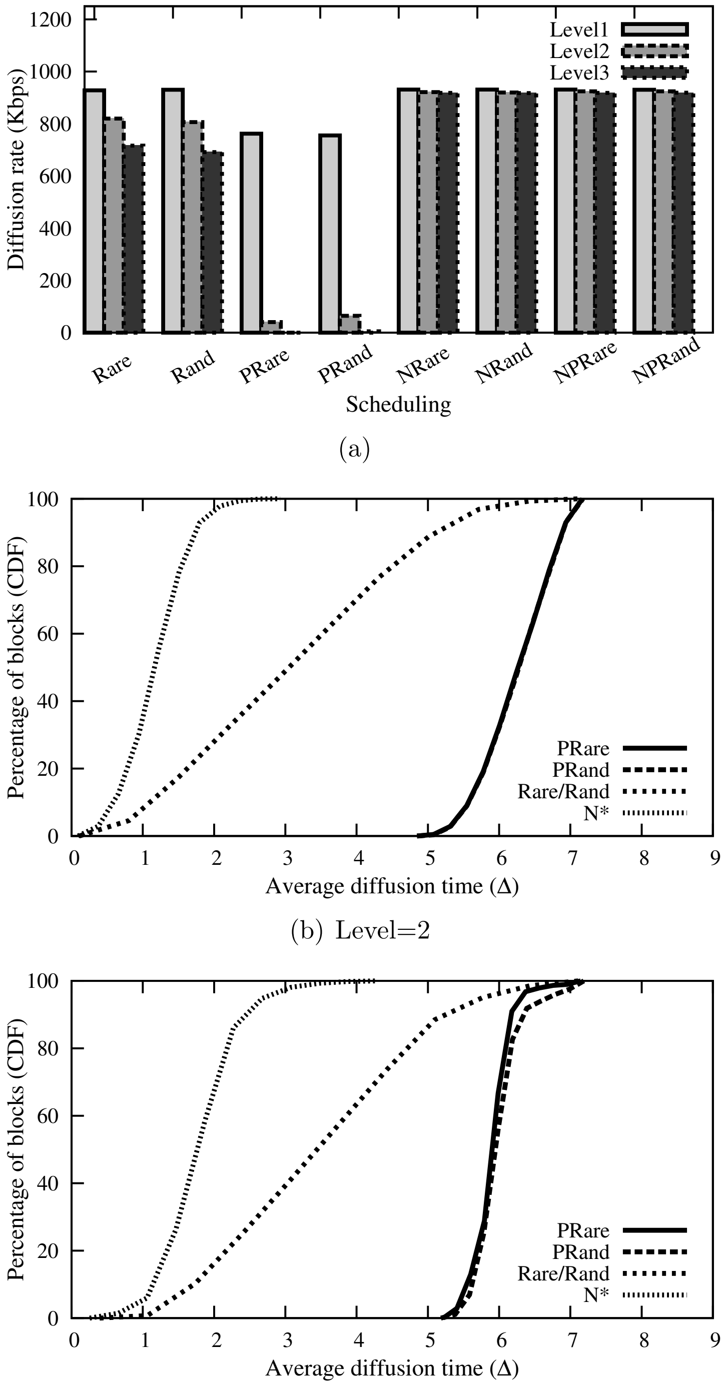
Figures 44.(a) shows the diffusion rate to the top three levels of the overlay for all the eight candidate scheduling schemes when peer degree is 4. Figures 44.(b) and 44.(c) present the distribution of diffusion time across all delivered blocks to top two levels (in terms of the number of intervals Δ) for peer degree 4. Sub-figures in 45. show the same results for peer degree 10. Overall these reveal that peer degree does not have a major impact on the diffusion rate and time in *NP**, *N** and *Rare* or *Rand* schemes. However, peer degree affects the diffusion time and rate of blocks in *PRare* and *PRand* schemes. Increasing the number of level 1 peers magnifies the role of source coordination. Each level 1 peer has a fixed block-budget to request from source, increasing number of peers in level 1 (source children) reduces block-budget to each one of them while keeps the aggregate budget fixed. Therefore, the probability of requesting redundant blocks specially from playing window increases which results in decrease of diffusion rate to level 1.

The rest of the results show similar trend which approve our previous findings for degree 6 and reveals that regardless of peer degree our findings are valid.

5.5.3. Bi-directional Overlay

Now, we examine the effect of bi-directional overlays on the performance of the candidate scheduling schemes.

Figure 46.(a) shows the diffusion rate to the top three levels of the overlay for all the eight candidate scheduling schemes when overlay is bi-directional. Figures



(a)

(b) Level=2

(c) Level=3

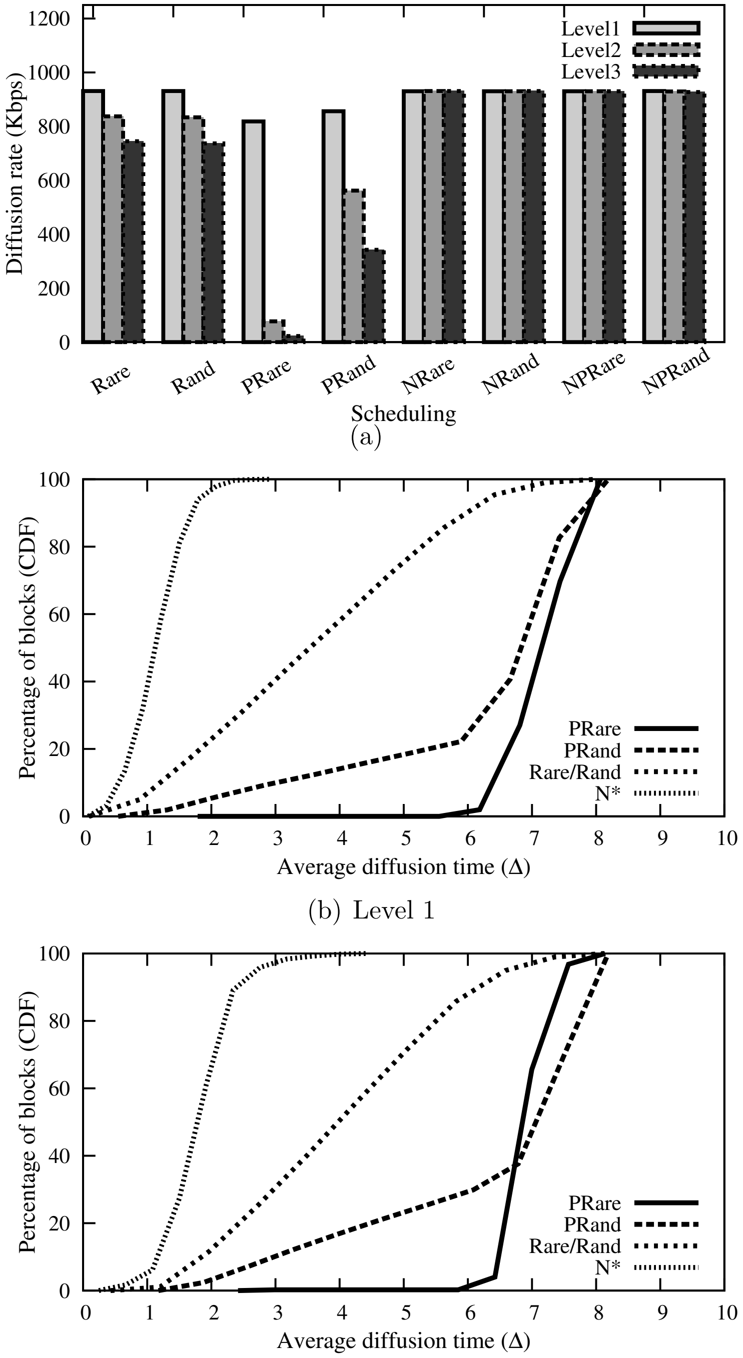
FIGURE 45.: Effect of peer degree: (a) Diffusion rate across different levels for various schedulings with peer degree 10. (b) and (c) Distribution of diffusion time for various schedulings with peer degree of 10 across level 1 and 2 peer, respectively.

46.(b) and 46.(c) present the distribution of diffusion time across all delivered blocks to top two levels (in terms of the number of intervals Δ) for a bi-directional overlay. We can observe that bi-directional overlay does not have any impact on diffusion time and rate to different levels of the overlay for various scheduling schemes.

Figure 47.(a) shows the distribution of bandwidth utilization for various scheduling schemes in a bi-directional overlay. This figure reveals the same trend in the performance of various schemes. Moreover, by comparing figures 38.(a) and 47.(a) we can observe that in a bi-directional overlay regardless of scheduling schemes utilization of bandwidth decreases. Figure 47.(b) presents the percentage of available blocks among parents across different windows for various scheduling schemes in a bi-directional overlay. This figure also confirms our previous findings about diversity of blocks among parents for different scheduling schemes. Clearly, a comparison between this figure and Figure 40.(a) reveals that in a bi-directional overlay diversity (percentage of available blocks among parents) regardless of scheduling schemes decreases. The rest of the results show similar trend which approve our previous findings for uni-directional overlays and reveals our findings are still valid in bi-directional overlays.

5.5.4. Bandwidth Heterogeneity

To explore a more realistic scenario, we now examine how heterogeneity of peers' bandwidth and percentage of high bandwidth peers affects the performance of various scheduling schemes in the resource constraint environment. We consider an

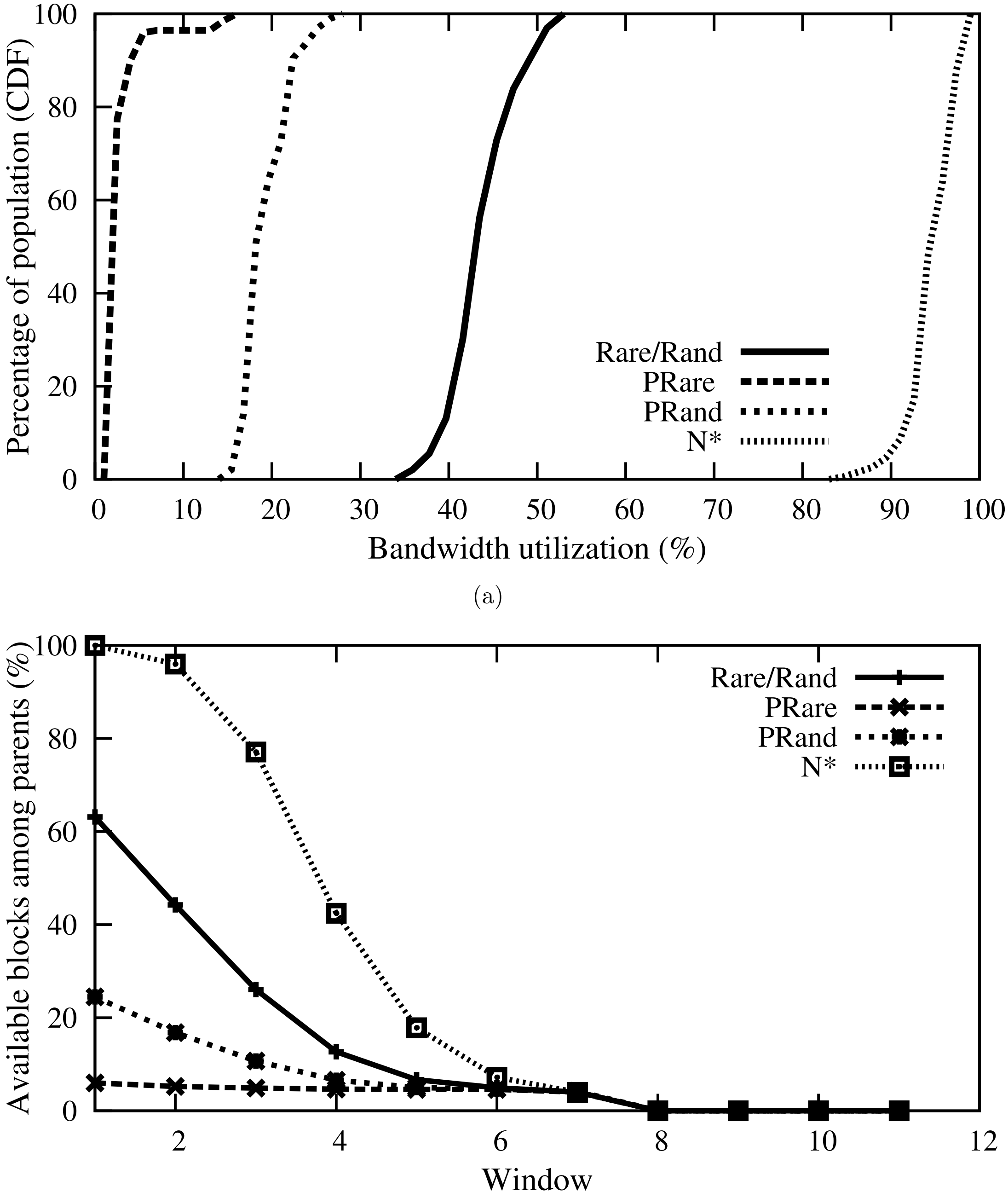


(a)

(b) Level 1

(c) Level 2

FIGURE 46.: Effect of bi-directional overlays: (a) Diffusion rate across different levels for various scheduling schemes. (b) and (c) Distribution of diffusion time for various scheduling schemes across level 1 and 2 respectively.



(a)

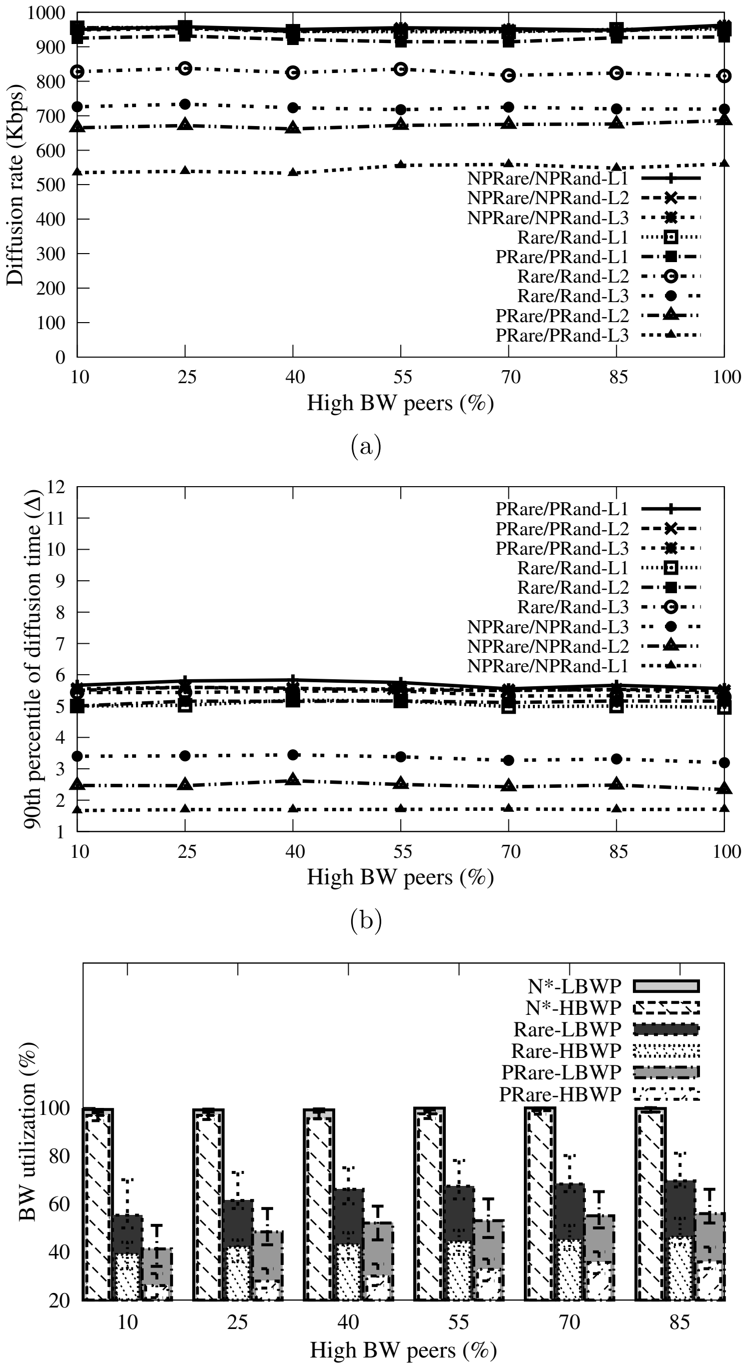
(b)

FIGURE 47.: Effect of bi-directional overlays: (a) Distribution of utilization of bandwidth across various scheduling schemes. (b) Percentage of available blocks among parents across different windows for various scheduling schemes.

overlay with two groups of peers with symmetric access link bandwidth of 480 Kbps (3 descriptions) and 960 Kbps (6 descriptions), and peer degree of 6 and 12, respectively. All other parameters are similar to the homogeneous scenario. Figures 48.(a) and 48.(b) depict the average diffusion rate and 90th percentile of block diffusion time to the top three levels for different scheduling schemes as a function of the percentage of high bandwidth peers in the group, respectively. The results for $NRare/NRand$ schemes are very similar to *NP** and are not shown for the clarity of figures.

Overall the results of different scheduling schemes are qualitatively similar to the homogeneous scenario. More specifically, *N** scheduling schemes achieve maximum diffusion rate and minimum block diffusion time across all levels of the overlay, regardless of the percentage of high bandwidth peers. *P** scheduling schemes request portion of the blocks in playing region from source which leads to late arrival of these blocks to level 1, and thus decreases the diffusion rate to lower levels. $Rare/Rand$ suffer from the same problem but perform slightly better than *P** schemes as we described in Subsection 5.5.1..

To examine the effect of bandwidth heterogeneity in each group of peers, Figure 48.(c) depicts the median, 10th and 90th percentile of incoming bandwidth utilization of high and low bandwidth peers (denoted as HBWP and LBWP). While the relative performance of different scheduling schemes within high and low bandwidth peers are similar, the utilization of high bandwidth peers is around 20% lower than low bandwidth peers in *P** and $Rare/Rand$ schemes. The reason is due to the lower ratio



(a)

(b)

(c)

FIGURE 48.: Effect of bandwidth heterogeneity: (a) and (b) Diffusion rate and 90th percentile of diffusion time, as a function of the percentage of high bandwidth peers, respectively. (c) Median, 10th and 90th percentile of bandwidth utilization for high and low bandwidth peers, as a function of percentage of high bandwidth peers.

of bottom level diffusion rate to high bandwidth peers' incoming bandwidth compared to the ratio for low bandwidth peers ($e.g.$, $\frac{700}{960}$ vs. $\frac{700}{480}$ in $Rare/Rand$). While the diffusion rate to all levels of the overlay in $Rare/Rand$ and $P*$ schemes (700 and 510 Kbps) is higher than the required quality for low bandwidth peers (480 Kbps) as shown in Figure 48.(a), even low bandwidth peers do not achieve high utilization, $i.e.$, can not receive their maximum quality. This is due to the long block diffusion time for different levels of the overlay which allows only half of the delivered blocks to the bottom level (that arrive within 4 intervals) to have sufficient time for effective swarming with minimum buffer size ($\omega = 7$). For instance, in *Rand*/*Rare* scheme, the diffusion rate to the bottom level is 700Kbps which leads to $(50\% * 700Kbps)/480Kbps$ or roughly 72% utilization for low bandwidth peers as shown in Figure 48.(c).

Note that the bandwidth utilization or delivered quality for individual peers depend on the aggregate quality of available content among their parents. Therefore, increasing the percentage of high bandwidth peers improves the availability of content among parents of each peer by increasing the probability of having a high bandwidth parent. This leads to some improvement in the utilization of both groups of peers by providing more diversity among blocks and facilitating a better swarming, as shown in Figure 48.(c).

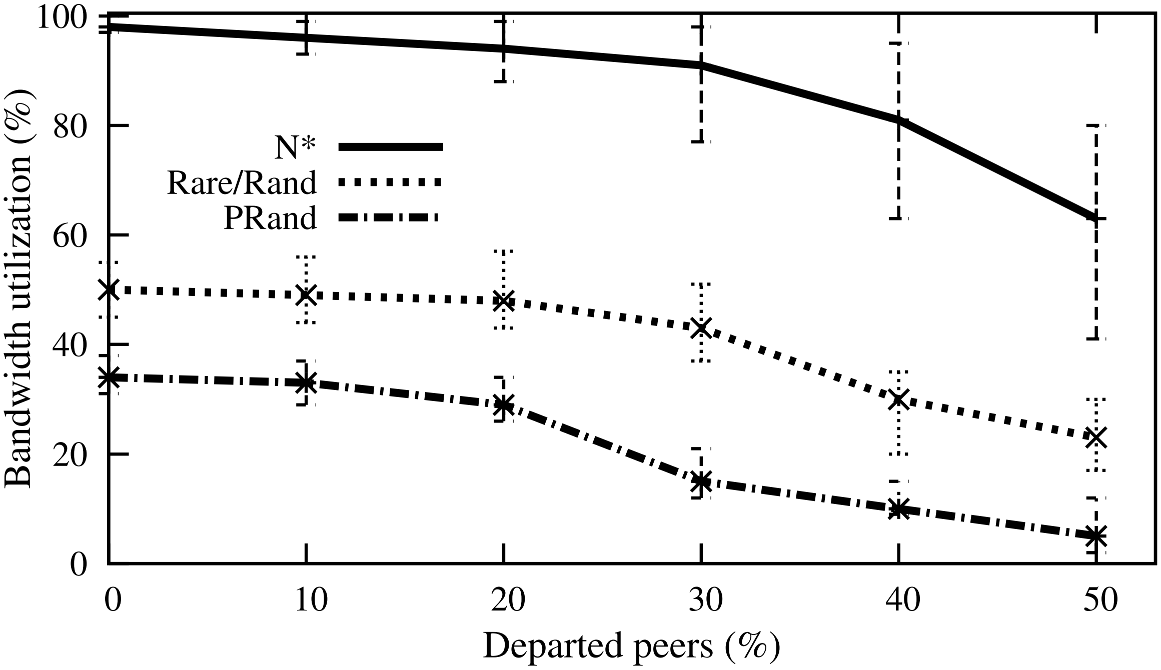
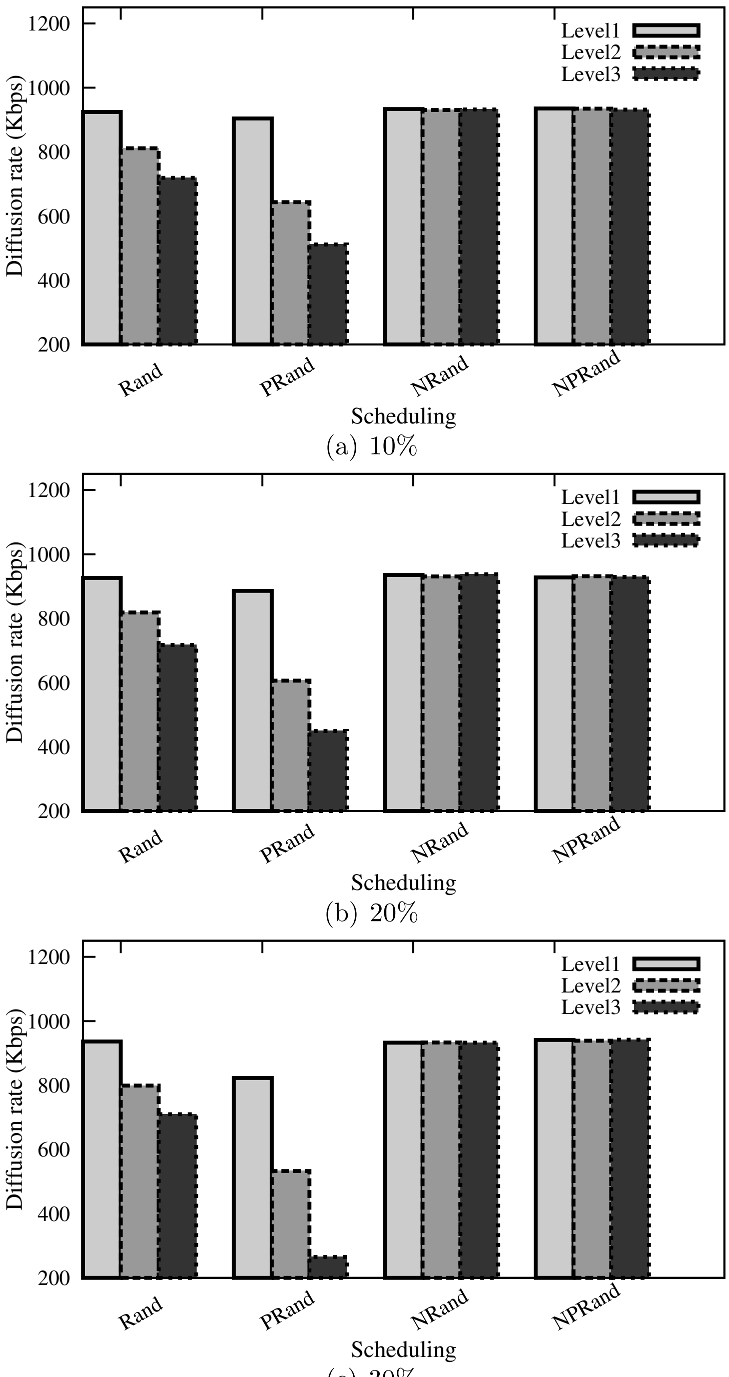


FIGURE 49.: Effect of distorted overlay: Percentage of utilization of bandwidth across various scheduling schemes for different percentage of departed peers.

5.5.5. Distorted Overlay

In this subsection, our goal is to investigate the ability of various scheduling schemes to cope with a distorted overlay. We model a distorted overlay by removing a random subset of participating peers from a properly connected overlay without repairing it. The performance of the content delivery in such distorted overlays represents the performance in the worst case scenario in presence of churn/peer-departure.

Figure 49. shows the median, 5th and 90th percentile of percentage of bandwidth utilization for various scheduling schemes across different levels of distortion. Clearly



(a) 10%

(b) 20%

(c) 30%

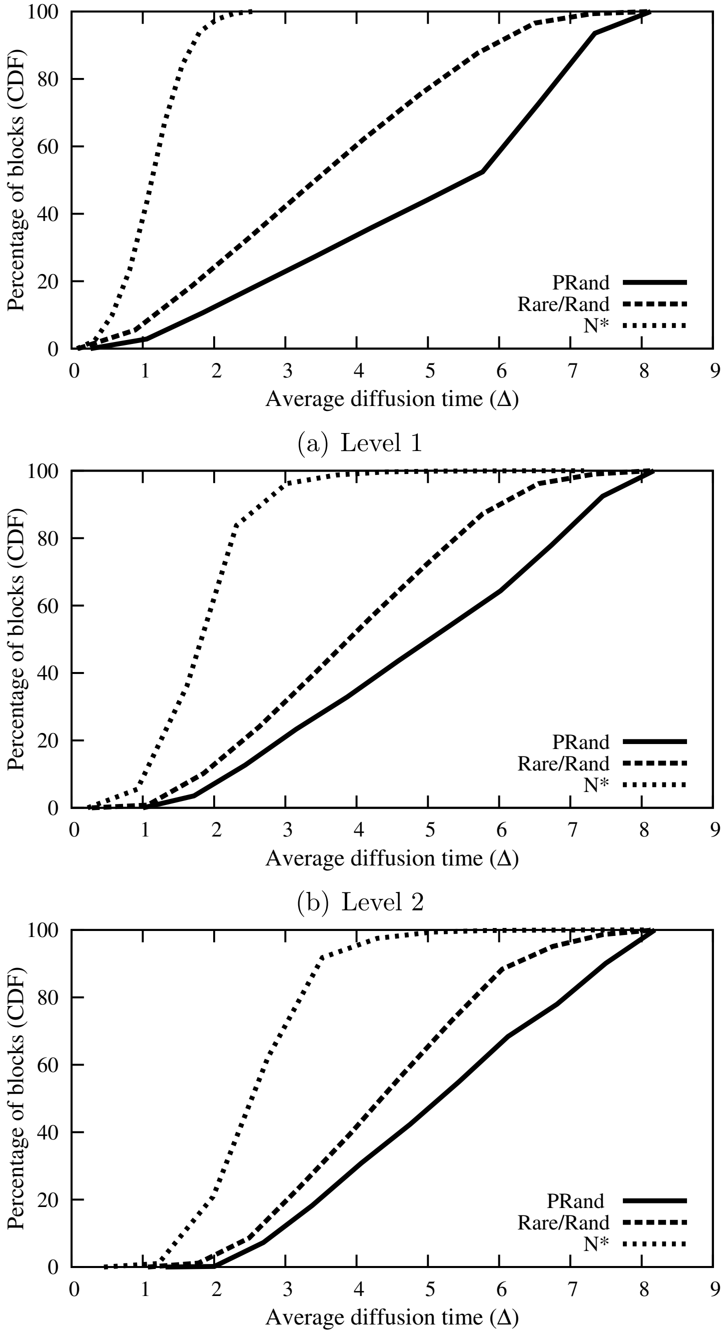
FIGURE 50.: Effect of distorted overlay: (a), (b) and (c) Diffusion rate across different levels for various scheduling schemes when 10%, 20% and 30% of peers left the overlay, respectively.

by increasing the distortion bandwidth utilization decreases. Figure 49. reveals that all of the scheduling schemes show the same trend.

Figures 50.(a), 50.(b) and 50.(c) show the diffusion rate to various levels for different scheduling schemes after the departure of 10%, 20% and 30% of peers from the system, respectively. Figures 51.(a), 51.(b) and 51.(c) depict the distribution of diffusion time across all delivered blocks to top three levels (in terms of the number of intervals Δ) for 10% of distortion. Figures 52. and 53. show the same set of results for 20% and 30% distortion in the overlay. From these figures we can observe that except for *PRare* the behavior of various scheduling schemes does not change by different levels of distortion in the overlay. *PRare* exhibits worse performance by increase in the distortion as the majority of block becomes available at the first level very late which can be seen from the diffusion time.

5.5.6. Peer Dynamics

The dynamics of peer participation (or churn) could disrupt content delivery and adversely affect the delivered quality to peers for two reasons as follows: *(i)* Departure of a direct parent, or any upstream peer along the diffusion path from source could affect the delivery of content to individual peers. In particular, in a resource constraint scenario where source delivers only a single copy of each block to the system, immediate departure of a level one peer that received the block prevents further diffusion of such a block to the rest of the overlay. This could significantly

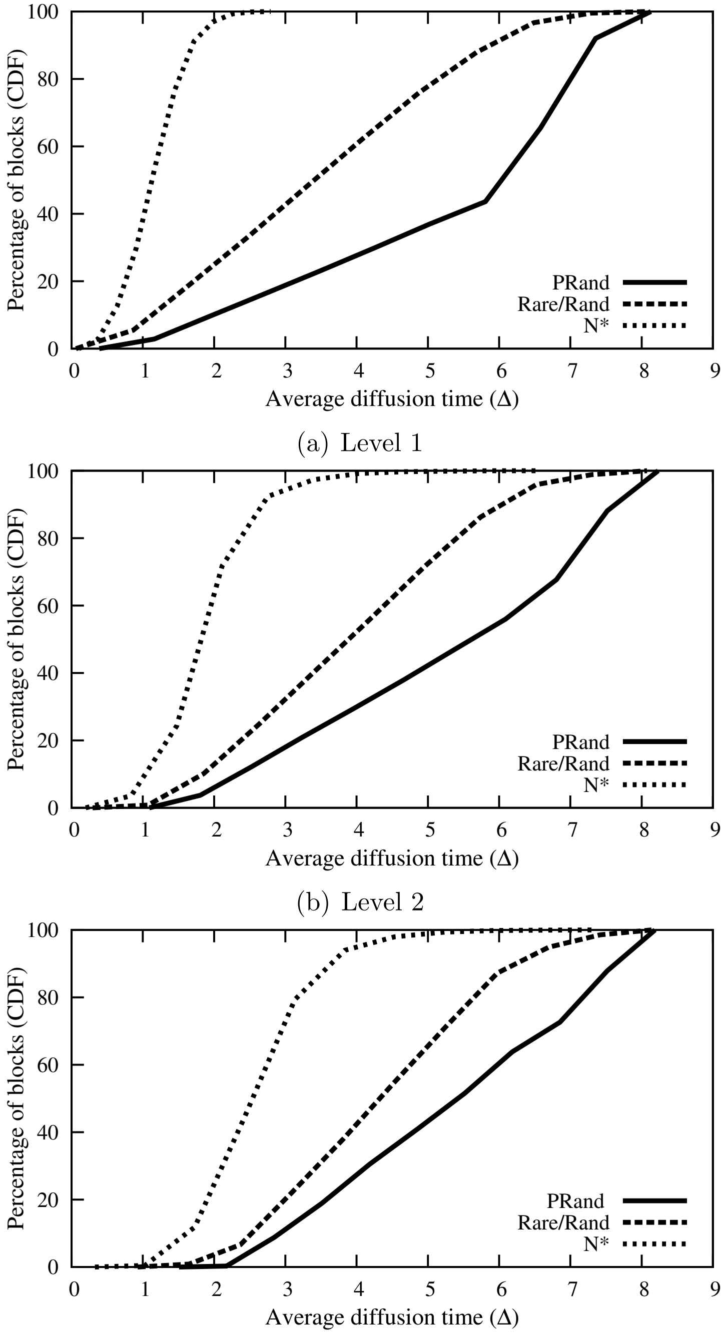


(a) Level 1

(b) Level 2

(c) Level 3

FIGURE 51.: Effect of distorted overlay: (a), (b) and (c) Distribution of per block diffusion time for various scheduling schemes when 10% of peers left the overlay across level 1, 2 and 3, respectively.

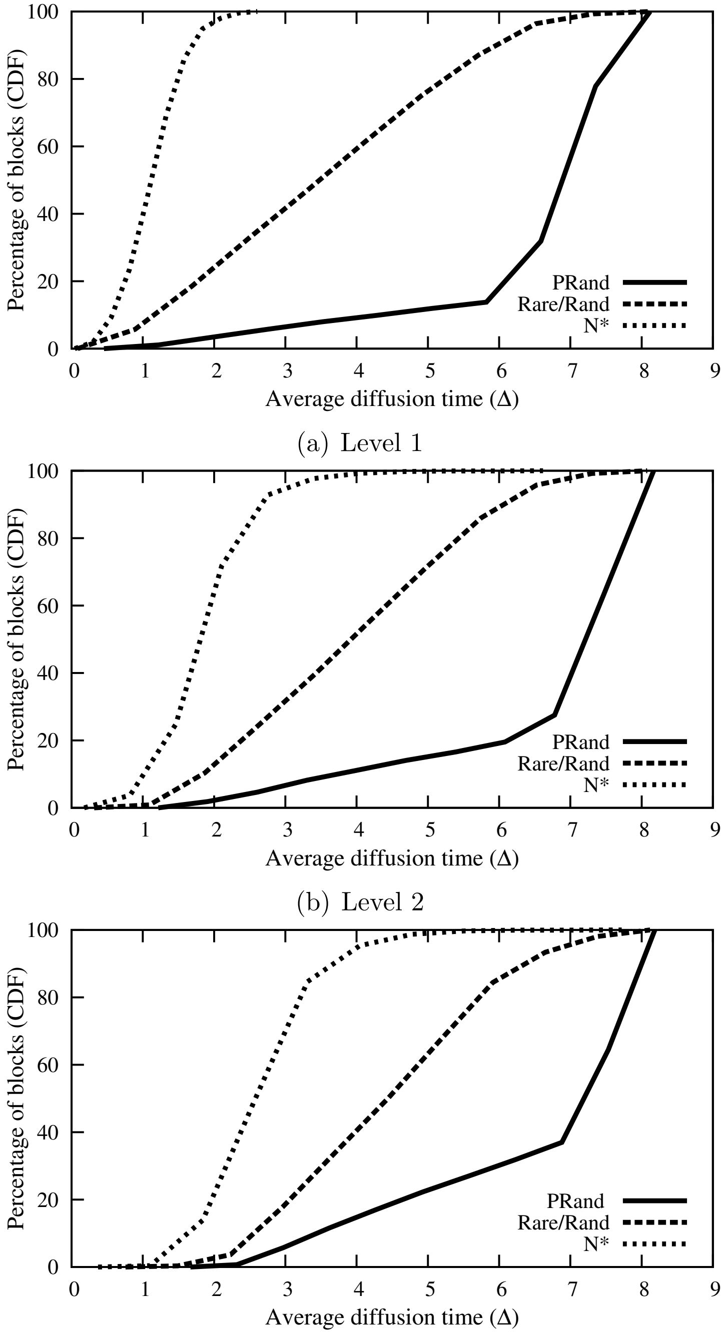


(a) Level 1

(b) Level 2

(c) Level 3

FIGURE 52.: Effect of distorted overlay: (a), (b) and (c) Distribution of per block diffusion time for various scheduling schemes when 20% of peers left the overlay across level 1, 2 and 3, respectively.



(a) Level 1

(b) Level 2

(c) Level 3

FIGURE 53.: Effect of distorted overlay: (a), (b) and (c) Distribution of per block diffusion time for various scheduling schemes when 30% of peers left the overlay across level 1, 2 and 3, respectively.

decrease the diffusion rate to lower levels. *(ii)* A subset of peers may have less than their target number of parents which in turn affects their aggregate incoming bandwidth and thus may degrade their delivered quality. This could also have an impact on their children.

In this subsection, we examine the effect of churn on the performance of $N*$, $PRare$ and *Rare* scheduling schemes. The schemes with random selection exhibit a very similar behavior and thus are not shown.

To incorporate a realistic model for churn, we select peer session times from a log-normal distribution (μ=4.29 and σ=1.28) and peer inter-arrival times from a Pareto distribution (a=2.52 and b=1.55) which is reported by empirical studies [131, 87]. The length of each simulation is 6000 seconds. We look at the result in heterogeneous and resource constraint scenario with two groups of peers with symmetric bandwidth of 960 and 480 Kbps with degree of 12 and 6, respectively. Source bandwidth is set to 1 Mbps.

In the presence of churn, a peer may leave the system shortly after receiving the first copy of a block at a particular level of the overlay. Such a peer is unable to pass any newly arrived block to its children in lower levels. This means that the definition of diffusion rate for static scenario would over-estimate the effective diffusion rate in the system. To address this issue, we slightly revise the calculation of diffusion rate as follows: when a peer in level i leaves the system, the blocks that it has received

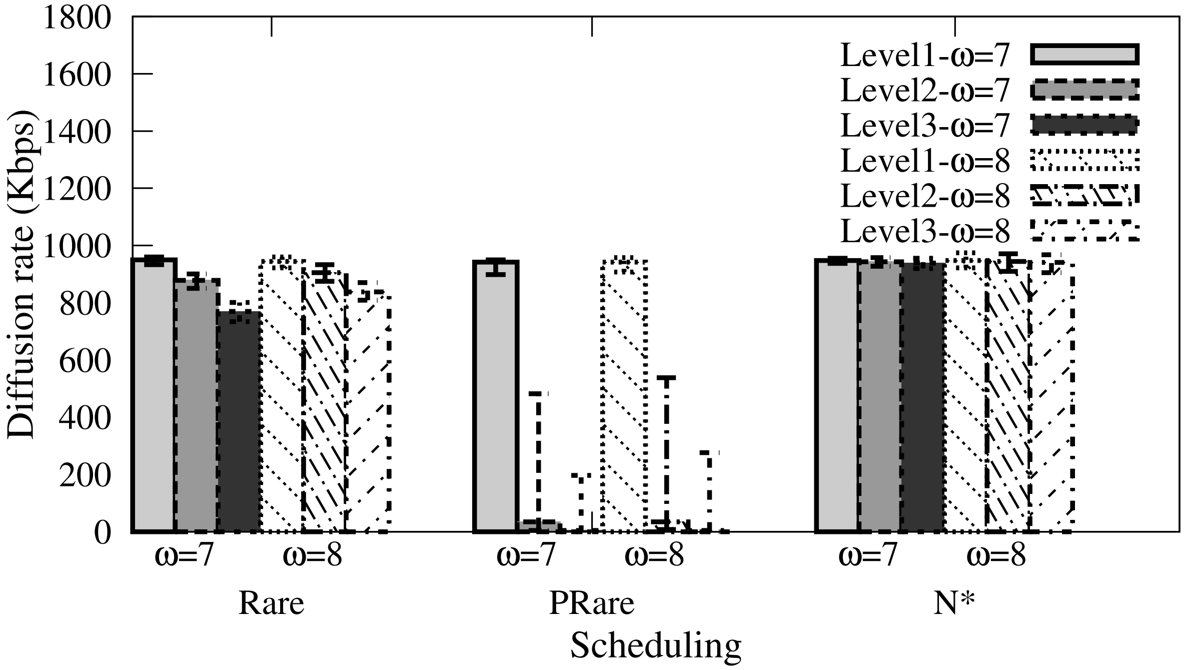


FIGURE 54.: Effect of churn: 50th percentile of diffusion rate of different levels along with 90 and 10-percentile as bars for various scheduling schemes.

during the last Δ seconds, are not considered in the calculations of the diffusion rate and diffusion time for level i.

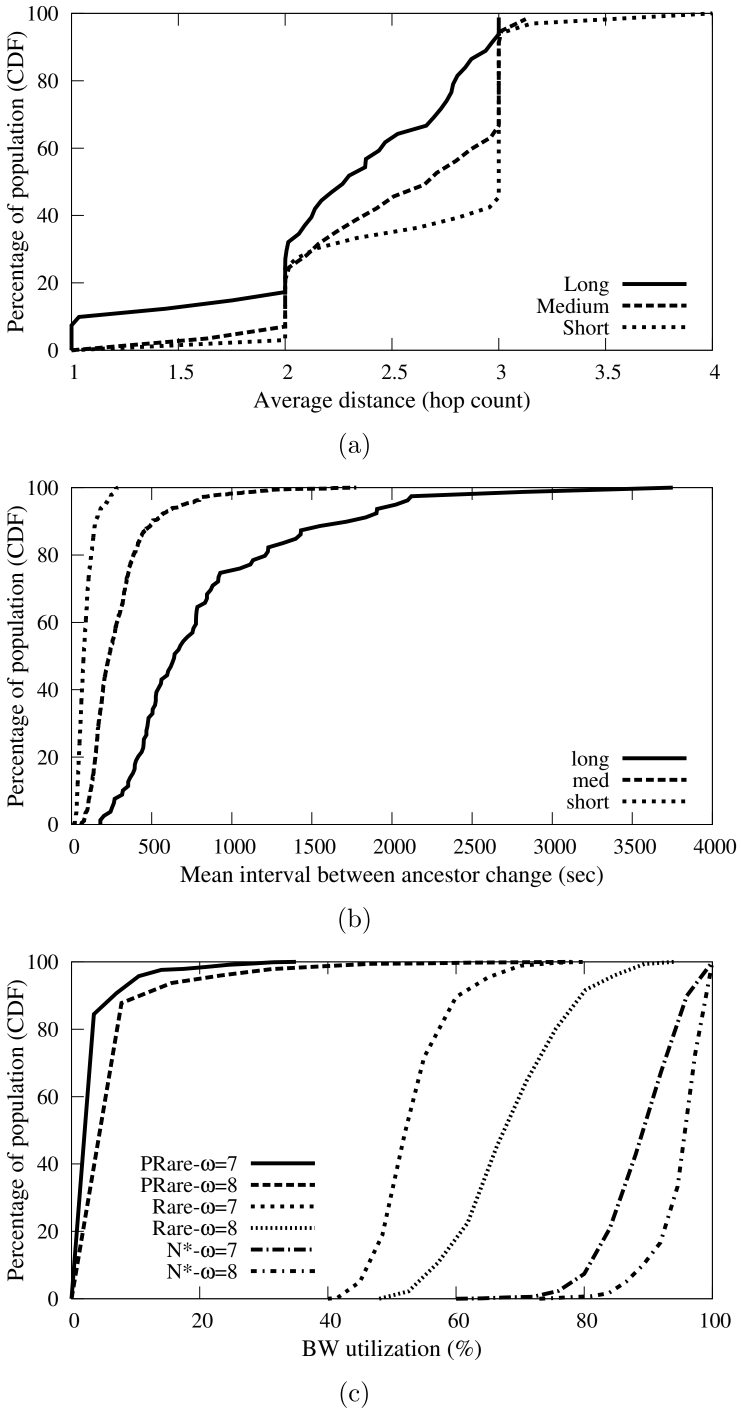
Figure 54. depicts the mean, 10th and 90th percentile (as bars) of diffusion rate for the top three levels of the dynamic overlay for the scheduling schemes. This figure illustrates that the diffusion rate for $N*$ and *Rare* schemes is roughly similar to the static scenario. However, the diffusion rate to lower levels is significantly lower than the static scenario for $PRare$ scheme. Results for the diffusion time are similar to static scenario (in figure 48.(b)) and are not shown.

To investigate the underlying causes of the impact of peer dynamics on diffusion rate, we examine the effect of churn on the connectivity among peers. We have noticed that while all peers experience the same mean time between consecutive disconnections of their parents, there is a correlation between peer uptime and the stability of its shortest path from source through the overlay (*i.e.*, the path for diffusion of blocks).

We divide all peers into three groups based on their session times (st) as follows: *(i)* short-lived where $st < 5$min, *(ii)* medium-lived where $5 < st < 30$min and *(iii)* long-lived where $st > 30$min. Figure 55.(a) illustrates the average shortest distance of peers from source for each one of the above three groups. This figure shows that long-lived peers tend to move to the higher levels of the overlay which improves the stability of the higher levels. The reason is that once a connection is established between two long-lived peers, it remains in place for a long period of time. This enables long-lived peers to gradually move to higher levels of the overlay and improves the stability of higher levels which affects the diffusion rate to lower levels. A complementary view for stability of the shortest path from source is the mean interval between departure of ancestor peers on the shortest path from source that is shown in Figure 55.(b). This figure reveals that peers with higher session times on average experience a higher degree of stability in their shortest path to the source. *Existence of these stable paths from source to each level explains the marginal impact of churn on the diffusion rate of* $N*$ *and* $Rare$ *schemes.* However, in $PRare$ scheme, when peers in level 1 lose their direct parents (in lower levels), they will have larger number of missing blocks in their playing region that are requested from source. This in turn increases the fraction of late blocks to level 1 and adversely affect the diffusion rate for lower levels.

Figure 55.(c) shows the distribution of incoming bandwidth utilization among high bandwidth peers for the same scenario. This figure reveals that while the diffusion rate of $N*$ and *Rare* are not significantly affected by peer dynamics, the average utilization of incoming bandwidth is slightly dropped compared to the static scenario (Figure 48.(c)). This is primarily due to the occasional departure of direct parents which triggers the peer discovery mechanism for identifying new parents. This further affects effective swarming and degrades the utilization of incoming bandwidth. To cope with the drop in bandwidth utilization, individual peers can increase their buffer size by one interval to facilitate effective swarming by providing a larger window to request any missing block. Figure 54. and 55.(c) show the corresponding results when peers use a larger window ($\omega = 8$). These figures show that using a larger buffer improves the utilization of bandwidth for *Rare* and $N*$ schemes as expected but has a minor impact on the $PRare$ scheme. The reason is that increasing buffer size does not affect the diffusion time and thus the percentage of late blocks to level 1 in the $PRare$ scheme.

To summarize, our results in this section, collectively illustrate that there is no qualitative and significant difference between static and dynamic environments in



(a)

(b)

(c)

FIGURE 55.: Effect of churn: (a) Distribution of average shortest distance of each peer from source for three groups of peers based on their session time. (b) Distribution of mean interval between shortest path ancestor change for three groups of peers based on their session time. (c) Distribution of incoming BW utilization.

terms of performance of various scheduling schemes. Moreover, diffusion rate (and time) is not affected by churn due to the stability of the higher levels of the overlay mesh which is the direct result of random construction of the overlay as shown in previous measurement studies on real P2P networks [132].

5.6. Performance Evaluation: Resources

We now turn our attention to the effect of excess resources, *i.e.*, source and peer bandwidth. We focus on *Rare* and *Rand* scheduling schemes that perform poorly in the resource constraint scenario, and examine how the availability of more resources affects their performance. More specifically, we investigate both the independent and combined effect of excess source and peer bandwidth on system performance by quantifying their impact on diffusion rate and diffusion time across levels of the overlay. We only show the results for *Rare* scheme. The results of *Rand* are very similar. We use minimum buffering in these scenarios to eliminate any side effect of buffer on our analysis. The effect of buffer size is later examined in this section. We consider a heterogeneous scenario with symmetric access link where half of the peers have 960 Kbps and the other half have 460 Kbps bandwidth with peer degree of 10 and 5, respectively. All other parameters are set to their default values. As we increase source bandwidth, its degree (number of source children) proportionally increases so that it has the same bandwidth to degree ratio as participating peers, *i.e.*, $\frac{bw}{deg}$ is $\frac{960}{10}$.

To minimize the effect of overlay connectivity on our results, we keep the peer connectivity constant across these scenarios. This implies that increasing source bandwidth proportionally increases source degree (*i.e.*, number of peers that directly connect to source) whereas increasing peer bandwidth has an opposite effect and proportionally decreases source degree[3].

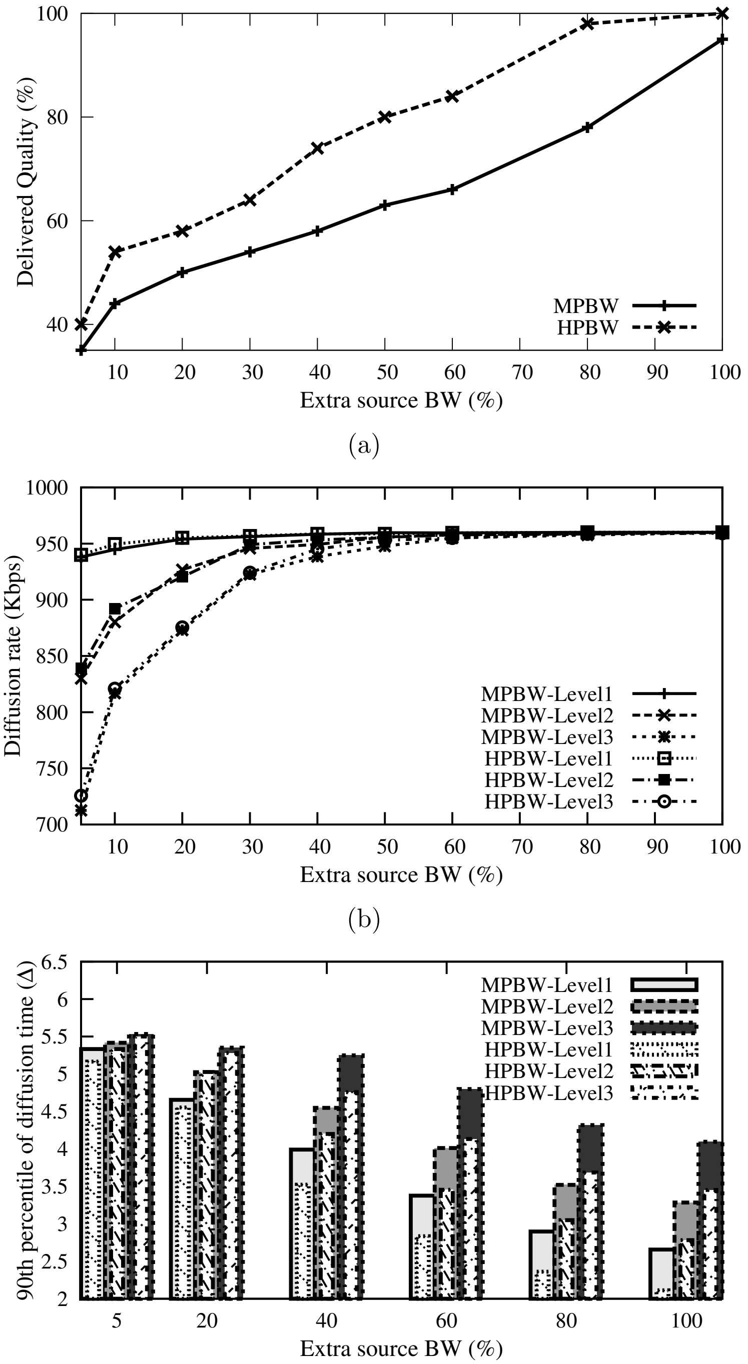
5.6.1. Effect of Source Bandwidth

We increase source bandwidth and adjust its out degree accordingly. Peer bandwidth is kept as minimum (*i.e.*, equal to the stream rate). We call this scenario MPBW. Figure 56.(a) demonstrates the 5th percentile of delivered quality to all peers as a function of the extra source bandwidth (only the line labeled MPBW) for *Rare* scheduling scheme. This figure reveals that increasing source bandwidth improves the performance by increasing the delivered quality to individual peers. The good performance in which 95 percentage of peers receive 95% of quality or more can be achieved when the source bandwidth is increased by 100%. To explain the observed performance of *Rare* scheduling scheme as a function of extra source bandwidth we investigate the diffusion rate and time. Figures 56.(b) and 56.(c) (only lines that

[3] We note that any change in the degree of source directly affect the depth of the overlay and changes the required buffering at each peer. However, since our methodology focuses on the pattern of delivery through levels, these changes can be captured by our methodology and does not affect our discussion.

labeled *Level i*) depict the diffusion rate and 90th percentile value of diffusion time for top three levels, respectively, as a function of extra source bandwidth. Figure 56.(b) clearly illustrates that the diffusion rate to level 2 and 3 rapidly increases with source bandwidth until they reach to the stream bandwidth (*i.e.*, 960 Kbps). Figure 56.(c) reveals the underlying reason for the increase in diffusion rate. Increasing source bandwidth directly decreases the diffusion time of blocks to level 1, which in turn reduces the fraction of late blocks to level 1 and allows peers in level 2 to pull more blocks from peers in level 1. This has a ripple effect on the diffusion time of blocks to lower levels and similarly increases the diffusion rate of those levels as well.

Note that the aggregate bandwidth for the delivery of new blocks to level 1 is limited by source bandwidth which is the main determining factor on the diffusion rate for level 1. However, the aggregate bandwidth between other levels is proportional to the number of connections between them (or roughly the number of peers in the lower level) which is considerably large. For example, in a scenario with peer degree 6, the maximum number of connections to level 1, 2 and 3 are 6, 36 and 216 connections, respectively. Therefore, the main performance bottleneck for diffusion rate to different levels, except level 1, is the diffusion rate and diffusion time for level 1 (as opposed to the aggregate bandwidth between lower levels). This observation explains the faster increase in the diffusion rate of lower levels with source bandwidth. More specifically, as diffusion time of blocks to the top level decreases, the abundant



(a)

(b)

(c)

FIGURE 56.: Effect of resources: (a) Distribution of delivered quality as a function of the extra source bandwidth for MPBW and HPBW scenarios for *Rare* scheduling. (b) and (c) Diffusion rate and 90-percentile of diffusion time of different levels as source bandwidth increases for *Rare*.

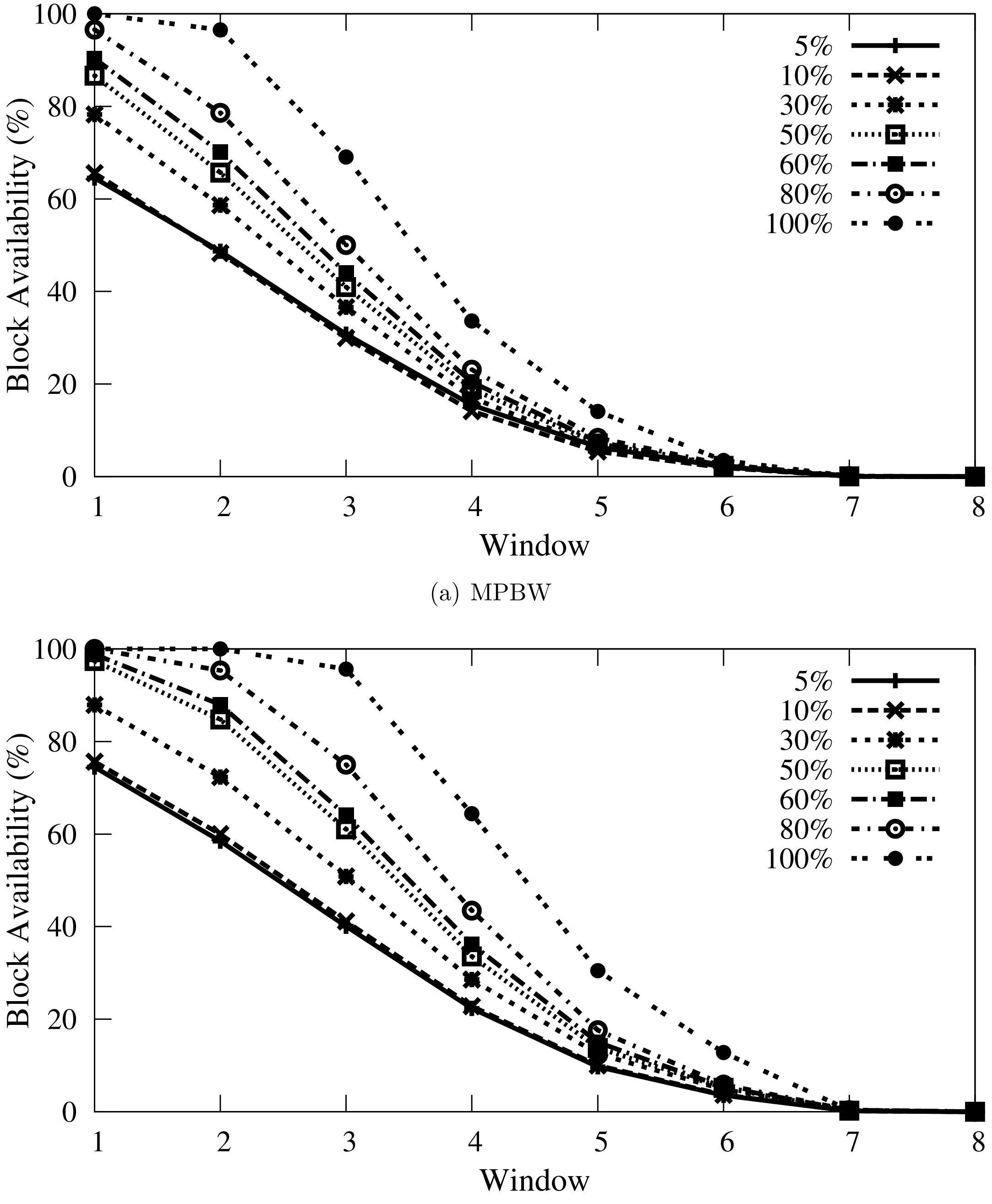
available bandwidth between lower levels can be utilized more effectively, and causes an even larger reduction in the diffusion time of the next level.

Figure 57.(a) depicts the block availability of level 2 peers for different extra source bandwidth. Comparing diffusion rate in Figure 56.(b) and block availability reveals despite reaching an adequate diffusion rate by 60% extra bandwidth, still the block availability is far from the optimal (observing $\frac{1}{deg}$ in the diffusion window and a significant increase in the swarming windows). This can be explained by longer diffusion time as shown in Figure 56.(c). Essentially, diffusion time with 60% extra source bandwidth is larger than Δ+1 intervals in level 1 and subsequently to lower levels. For example, in 60% extra source bandwidth, diffusion time to level 1, 2 and 3 peers is 3.5, 4.5 and 5.5. However, with N* schemes as shown in Figures 39.(a) and 39.(b), 90th percentile of diffusion time for top 2 levels is 1.7 and 2.7. Therefore, in *Rand* scheme, content becomes available approximately one Δ later in case of 60% extra source bandwidth and we observe the block availability starts from window 5 rather than 6 in Figure 57.(a). Further increase in source bandwidth does not change the diffusion rate across all levels of the overlay but decrease the diffusion time to level 1 as shown in Figure 56.(c). The improvement in diffusion time directly affects the block availability and provides enough block diversity for swarming within the give buffer.

5.6.2. Effect of Peer Bandwidth

Increasing peer bandwidth increases the aggregate bandwidth between levels but does not have any effect on the bandwidth from source to level 1. To examine the effect of peer bandwidth, we double peer bandwidth in the resource constraint scenario, and call this scenario high peer bandwidth (labeled as HPBW). The two data points on the y-axis of Figure 56.(a) show the 5th percentile of delivered quality to individual peers for MPBW and HPBW scenarios in which the source bandwidth is minimum (*i.e.*, 5% extra). This result reveals that doubling peer bandwidth while source bandwidth is minimum, increases the percentage of delivered quality from 35% to 40 %, which clearly is a very minor improvement.

The six data points on the y-axis of Figure 56.(b) present the diffusion rate of the three levels for both MPBW and HPBW scenarios. Our results illustrate that doubling peer bandwidth has a negligible effect on the diffusion rate or even diffusion time (shown in Figure 56.(c) under 5% extra source BW). This may seem surprising because increasing peer bandwidth for all peers significantly increases the aggregate resources in the system compared to the increasing source bandwidth alone. However, this result supports our earlier explanation, that the available bandwidth between levels is already abundant, and increasing peer bandwidth does not change the diffusion rate or diffusion time to level 1 which are the main performance bottleneck and are not affected despite this dramatic increase in available resources.



(a) MPBW

(b) HPBW

FIGURE 57.: Effect of resources: (a) and (b) Block availability in level 2 peers per window for various extra source bandwidth in MPBW and HPBW scenarios, respectively. The deployed block scheduling is *Rare*.

5.6.3. Combined Effect of Source and Peer Bandwidth

Figure 56.(a) depicts the 5th percentile of delivered quality for MPBW and HPBW scenarios as a function of the extra source bandwidth. This figure shows that increasing source bandwidth along with peer bandwidth improves the performance by increasing the delivered quality. In fact, doubling the peer bandwidth along with an extra 80% for source bandwidth result in 100% delivered quality for the majority of peers in *Rare* scheduling.

Figures 56.(b) and 56.(c) demonstrate the combined impact of source and peer bandwidth by showing the diffusion rate and time as a function of source bandwidth when peer bandwidth is doubled (*i.e.*, HPBW scenario). Figure 56.(b) shows that increasing peer bandwidth does not have any effect on the diffusion rate. However, comparing diffusion time in minimum peer bandwidth and HPBW scenario in Figure 56.(c), reveals that as source bandwidth increases, high peer bandwidth scenario can further drop the diffusion time to lower levels. To explain this, we recall that as source bandwidth increases, the delivered blocks to level 1 experience shorter diffusion time (as we have shown in Figure 56.(c)). This faster availability of new blocks to level 1 provides more time for delivery of blocks to other levels, and thus enables the system to deliver the same number of blocks within a shorter period of time without changing the diffusion rate to lower levels. In fact, short diffusion time of blocks to lower levels provides more time for effective swarming. This can be observed in Figure 57.(b) which is the block availability among parents of level 2 peers for HPBW scenario

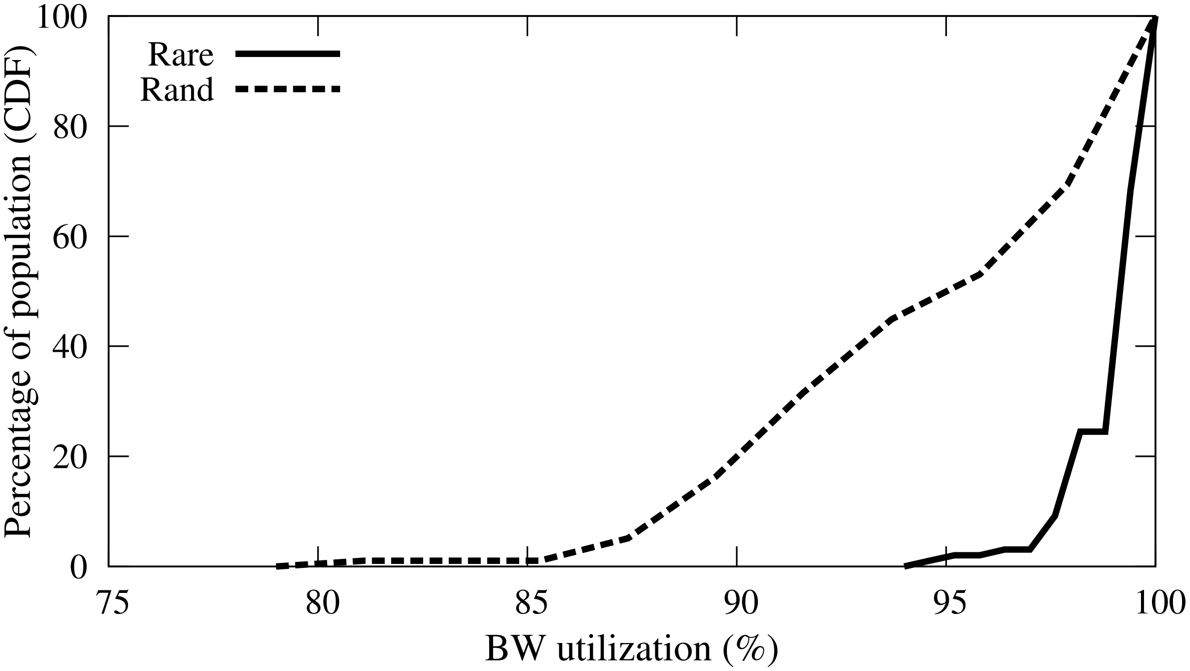


FIGURE 58.: Distribution of bandwidth utilization for *Rare* and *Rand* schemes. Source bandwidth is doubled.

as source bandwidth increases for *Rare* scheduling. Figure 57.(b) reveals that the availability horizon of level 2 peers is window 6 while in case of MPBW in Figure 57.(a) it is window 5. The increase in the availability horizon provides more room for the swarming and thus the block availability in windows of 1 to $\omega - depth$ (*i.e.*, $7-4$ or 3 in this example) significantly increases. Essentially, decrease in diffusion time, stretches the horizon of available blocks among parents of peers in all levels and thus provides more opportunities (windows) for effective swarming.

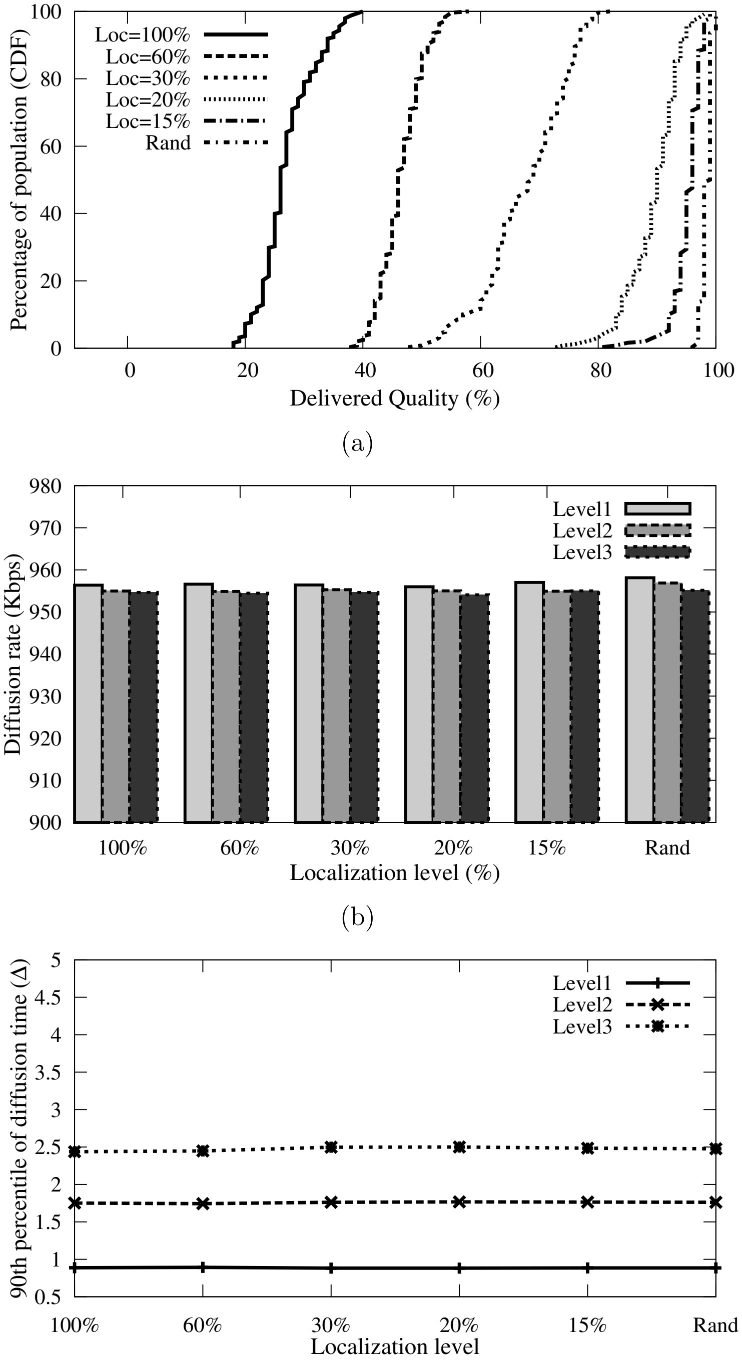
This in turn enables the system to operate with a smaller amount of buffering.

Taking a Closer Look: Our evaluation methodology effectively captures the global pattern of content delivery which represent the primary factors that affect the overall performance. However, there might be some minor differences between two scheduling schemes that are not detected by our metrics. Figure 58. depicts the distribution of bandwidth utilization for *Rare* and *Rand* schemes where source bandwidth is doubled. While all other characteristics of these two schemes were pretty similar, this figure indicates that peers experience a slightly better performance with *Rand* scheme when this excess resources become available. This illustrates the complexity of performance evaluation in mesh-based P2P streaming.

5.7. Performance Evaluation: Overlay Localization

Recently, there is a growing interest in localization of overlay connectivity within each edge ISP to limit the amount of inter-ISP traffic [111]. We focus on this design choice of overlay construction to demonstrate the abilities of our evaluation metrics in capturing the performance of mesh-based P2P streaming mechanisms over localized overlays.

We consider the heterogeneous scenario with 500 peers where 50% of peers are high bandwidth and the rest are low bandwidth. We consider a resource constraint setting. All the other parameters are set to their default values. We incorporate one of the best performed block scheduling scheme, namely *NPRare*. In the localized settings, peers are homogeneously distributed across 10 different ISPs. We define



(a)

(b)

(c)

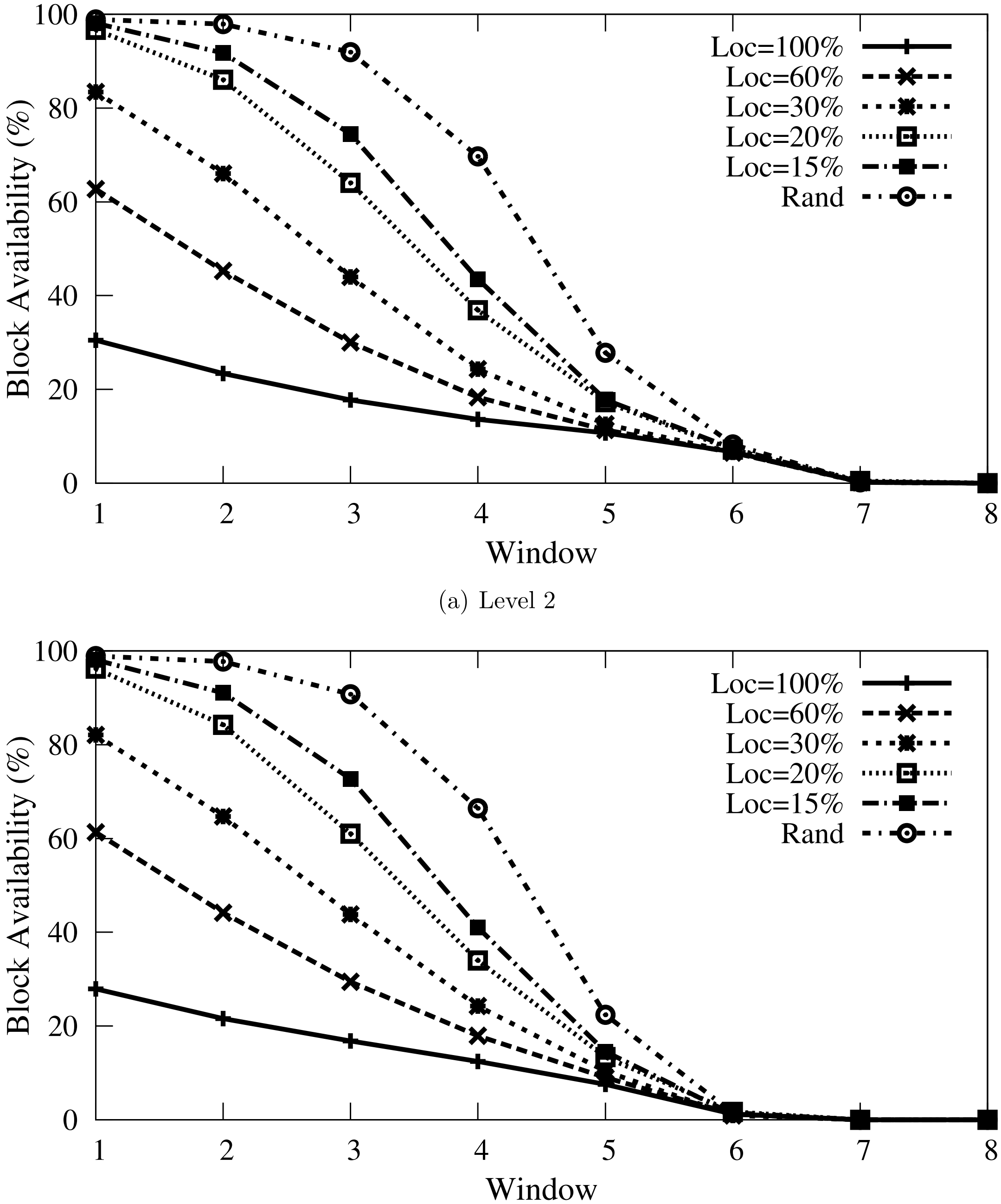
FIGURE 59.: Effect of overlay localization: (a) Distribution of delivered quality to peers across overlays with various levels of localization and a random overlay for comparison. (b) and (c) Diffusion rate and time of different levels of overlays with various levels localization and a random overlay, respectively.

the level of localization as the ratio of stream rate to aggregate external incoming bandwidth for each ISP. For each individual ISP, we control the level of localization or ISPs external incoming traffic based on the total number of external incoming connections to all peers in the ISP. For instance, in case of 100% localization, the aggregate external incoming bandwidth to each ISP is equal to the stream rate.

Figure 59.(a) depicts the distribution of the percentage of delivered quality to high bandwidth peers for overlays with various level of localization. The line represents by ‘Rand‘ is the delivered quality in a random overlay. Clearly this figure reveals that localization significantly degrades the delivered quality to individual peers, while decreasing the localization improves the performance.

To investigate the underlying reasons for the poor performance of content delivery in localized overlays, we present the diffusion rate and time. Figures 59.(b) and 59.(c) demonstrate the diffusion rate and 95th percentile of diffusion time to the top three levels of the overlay as a function of the localization level. Observing these two performance metrics reveals that the content fully diffuses through various levels of the overlay in a timely manner (comparable to the random overlay).

After discovering that the diffusion rate and time are optimal, we turn our attention to the block availability metric which reveals the diversity of content in each neighborhood for effective swarming. Figures 60.(a) and 60.(b) depict the block availability across various windows for overlays with different levels of localization and the random overlay for level 2 and 3 peers, respectively. These figures demonstrate



(a) Level 2

(b) Level 3

FIGURE 60.: Effect of overlay localization: (a) and (b) Block Availability across all windows for overlays with various levels of localization and a random overlay for level 2 and 3 peers, respectively.

that the block availability is much lower in localized overlay compared to a random overlay. This low availability of blocks explains the poor performance of peers in localized overlays and their performance improvement by decreasing the localization.

Figures 60.(a) and 60.(b) reveals that the diversity of blocks among parents in the swarming windows is significantly lower for localized overlays compared to the random overlay. In fact these results demonstrate that due to the strict localization inside ISPs, connectivity between different subtrees are limited thus the slope of progress of block availability in swarming windows is low. Relaxing the localization improves the diversity by allowing more connections between ISPs and thus distinct diffusion subtrees.

Closer examination of Figures 60.(a) reveals for the diffusion window, which is the last window in these two figures (*i.e.*, 6th windows for level 2 peers), the block availability is similar across various level of localizations and the random overlay for level 2 peers. However, the last window for level 3 peers (Figure 60.(b)) experience lower block availability in localized overlays compared to the random overlay. The reason is that level 2 peers typically have one diffusion parent in level 1 from whom they will pull the diffusion content. Thus, $\frac{1}{deg}$ (*i.e.*, $\frac{1}{12}$ or 8%) of content is available in the diffusion window. Regardless of localization this amount of content is available in the diffusion window of the diffusion parent due to the optimal diffusion rate as shown in Figure 59.(b). However, in the random overlay the block availability in the diffusion window of level 3 peers is more than $\frac{1}{deg}$ while for localized overlays is roughly equal to $\frac{1}{deg}$. Recall that, due to the peer population and average incoming degree, peers in level 3 may have multiple diffusion parents in level 2. If these diffusion parents are residing in various diffusion subtree, the aggregate available content in the diffusion window among parents of level 3 peers will be more than $\frac{1}{deg}$. In fact, the amount of available blocks in the diffusion window is proportional to the number of level 2 diffusion parents belonging to distinct diffusion subtrees. Thus, we observe that block availability in the diffusion window of level 3 peers in the random overlay is more than 8% as shown in Figure 60.(b). On the other hand, in localized overlays, despite the fact that level 3 peers still have multiple diffusion parents, due to the strict localization most of their diffusion parents are belonging to the same diffusion subtrees. Therefore, the amount of distinct blocks available at the diffusion window does not increase from 8%.

5.8. Summary

In this chapter, we presented an effective methodology for performance evaluation of mesh-based P2P streaming mechanisms. Our methodology leverages the insights from PRIME and presents a set of metrics to capture the behavior of swarming content delivery. We utilized the proposed methodology to asses the performance of mesh-based P2P streaming solutions with eight different block scheduling schemes along with localized overlays in realistic scenarios that represent a wide range of mesh-based P2P streaming mechanisms.

Overall, our study provides a useful insight in the design of block scheduling schemes particularly, by revealing how different components of scheduling affect the global pattern of content delivery. Our evaluation methodology in essence offers a unified framework for head-to-head comparison of different block scheduling schemes. Furthermore, our findings provide useful guidelines for resource provisioning and stress testing of mesh-based P2P streaming systems.

Through our study, we reveal that despite the intuition that requesting blocks that are going to be played shortly by each peer is a good design choice, such a selfish scheduling approach results in a very poor performance in resource constraint settings. In fact, peers should look at the global goal by requesting blocks farthest away from their playtime which results in a good global pattern of delivery and maximizes the delivered quality to all peers in a scalable fashion. Our main findings through our study in this chapter can be summarized as follows:

- Scheduling schemes that prioritize pulling of newly available blocks with largest timestamps among parents exhibit a significantly better performance than other schemes. Only this class of scheduling schemes can achieve good performance in resource constraint scenarios. This implies that in mesh-based P2P streaming systems, the availability of new blocks is more important than addressing timing requirement.

- Increasing source bandwidth (with proper source coordination) results in a major improvement in performance compared to increasing peer bandwidth.

This is due to the unique role of source bandwidth on the rate and timing of delivered blocks to participating peers throughout the overlay. In contrast, increasing peer bandwidth has a limited effect on performance.

- Any poorly designed scheduling can provide good quality to participating peers by adding sufficient amount of excess resources of proper type and/or increasing buffer size.

In this dissertation, so far we have focused on the basic design of live P2P streaming mechanisms, their comparison and performance evaluation. Practical deployment of such mechanisms arises a set of issues that we address in the next chapters.

CHAPTER VI

INCORPORATING CONTRIBUTION-AWARENESS INTO MESH-BASED P2P STREAMING

Material in this chapter was adapted from a paper previously published [9] in the Journal of Peer-to-Peer Networking and Applications, July 2010. This work is co-authored by Prof. Reza Rejaie and Dr. Yang Guo. The experimental work is entirely mine. The writing is primarily mine, with small contributions from each of the co-authors, who also provided technical guidance.

The feasibility of P2P streaming primarily depends on the scalability of contributed (or available) resources, namely outgoing bandwidth, with the number of participating peers. In practice, the resources are often insufficient to maximize the delivered quality to individual peers [122]. Such a resource constraint scenario occurs when a subset of participating peers are unwilling or unable to contribute as much resources as they demand, and this deficit can not be compensated by the excess resources from other peers. Since allocated resources to individual peers determine their maximum delivered quality, in such a "resource constraint" setting, a fair scheme should allocate resources to individual peers proportional to their contributed resources (or their outgoing bandwidth) rather than their demand (or incoming bandwidth). Moreover, excess resources of high bandwidth peers should be divided to all peers proportional to their contribution. Allocating resources in a *contribution-aware* fashion provides incentives among participating peers to actively contribute their resources in order to improve their own observed quality. Incorporating contribution-awareness into P2P streaming is in essence, a distributed resource management problem which is challenging due to the distributed, heterogeneous and dynamic nature of available resources as peers join and leave the system. Because of these challenges, a majority of studies on P2P streaming have not directly address the issue of contribution-aware resource allocation.

In this chapter, we present a tax-based [95, 96] contribution-aware scheme for live mesh-based P2P streaming approach. Such a contribution-aware scheme is a promising approach to effectively manage distributed and dynamic resources in P2P streaming by controlling the connectivity (*i.e.*, number of parents) of individual peers. The effectiveness of incorporating tax-based contribution-aware scheme in the context of tree-based approach has been investigated by Sung et al. [95]. We focus on the mesh-based approach due to its superior performance and better scaling properties compared to the tree-based approach [5].

6.1. Contributions & Design Objectives

The goal of this work is to design a contribution-aware scheme for live mesh-based P2P streaming. Towards that we make three main contributions as follows: First, we describe how a tax-based contribution-aware scheme can be seamlessly incorporated into the mesh-based P2P streaming. Second, we examine the behavior of the proposed tax function in allocating the available resources among participating peers for different values of aggregate resources and tax rates. We identify the so-called "saturated region" where high bandwidth peers do not require their allocated share of resources, and examine the ability of the proposed scheme to effectively utilize these excess resources. Third, we perform extensive evaluations of the proposed contribution-aware scheme.

We show that the connectivity of individual peers directly determines their observed performance in the mesh-based P2P streaming. Then, to incorporate the contribution aware scheme in the context of mesh-based P2P streaming approach, each peer uses a given tax function to determine the number of parents it is "entitled" to (or its share of resources, *i.e.*, its incoming degree) based on the aggregate available resources in the system, and the amount of its contributed resources. To effectively utilize the excess resources in the system, the unsaturated peers can further increase their incoming degree by adaptively examining the possibility of increasing their number of parents. Each peer as a parent, incorporates a preemption policy to properly allocate its resources between existing and new child peers.

We investigate the effect of key design parameters (*e.g.*, tax rate and preemption policies) over a wide range of scenarios using a realistic model for peer dynamics and pairwise delay. In particular, we examine how changes in aggregate available resources, distribution of contributed resources among peers, and group size affect the allocation and overall utilization of resources, as well as the stability of the overlay.

6.2. Background

To design the contribution-aware scheme, we rely on PRIME as discussed in Chapter III as a representative mesh-based P2P streaming mechanism. While the description and discussions are centered around PRIME, we believe that most of the issues and findings are generally applicable to other mesh-based P2P streaming systems. The basics of PRIME is presented in Chapter III. In this section, we briefly present some background details that are required for this chapter.

To accommodate the bandwidth heterogeneity among peers and adapt the delivered quality accordingly, the video content can be encoded using either layered coding (LC) or multiple description coding (MDC). LC encodes the streaming content into one base and multiple enhancement layers. The base layer can be independently decoded, while the enhancement layers can only be decoded cumulatively. Sequential layers can be added/dropped to adapt the streaming quality [105, 106, 107]. LC is sensitive to data losses of lower layers (*i.e.*, without a base layer the video content cannot be decoded). On the other hand, MDC organizes the streaming content into several substreams where each substream can be independently decoded. The delivered quality is proportional to the number of received substreams. The choice of various coding schemes clearly impacts the system design where MDC allows for more design flexibility, and LC is better at coding efficiency. We believe that flexibility of MDC outweighs the efficiency of LC because it allows utilization of outgoing bandwidth of any parent regardless of which layer it has. Thus, we focus on MDC that allows each peer to receive the appropriate number of substreams that are delivered through its access link bandwidth and facilitates the incorporation of contribution awareness into PRIME[1].

6.2.1. Bandwidth-degree Constraint

To effectively utilize the access link bandwidth of peers, participating peers try to maintain their incoming and outgoing degrees proportional to their incoming (bw_{down}) and outgoing (bw_{up}) bandwidth as pointed out in Chapter III. Using the same ratio of incoming (or outgoing) bandwidth to incoming (or outgoing) degree for all peers implies that all connections have roughly the same average bandwidth which is called bandwidth-per-flow or $bwpf$. $bwpf$ is a configuration parameter that is selected a priori and known by individual peers. More specifically, each peer tries to maintain its incoming and outgoing degrees at $\lfloor\frac{bw_{down}}{bwpf}\rfloor$ and $\lfloor\frac{bw_{up}}{bwpf}\rfloor$, respectively.

[1] Note that, many previous studies have used this intuition to use MDC coding in their approach [87, 96, 95].

6.2.2. Controlling Allocated Resources

Since all connections have roughly the same bandwidth, the amount of resources (*i.e.*, bandwidth) that a peer contributes or consumes in the overlay can be approximated by its outgoing and incoming degree, respectively. More specifically, when the appropriate block scheduling and adequate buffer size at each peer are used, the delivered stream quality to each peer would be proportional to its incoming bandwidth [69]. To demonstrate this point, we conduct ns simulations where peers with heterogeneous and asymmetric access link bandwidth form a directed and randomly connected mesh. The MDC encoded streaming content has 10 descriptions and all descriptions have the same rate of 160 Kbps. The incoming access link bandwidth of all peers is set to 1.6 Mbps so that each peer can receive the full quality stream. The outgoing access link bandwidth of 20% of peers (high contributors) is set to 2.4 Mbps while for the rest of the peers (low contributors) is set to 400 Kbps. In this setting, the ratio of demand to supply for resource (or resource index) is 0.5. We evaluate the effect of $bwpf$ by setting the maximum incoming degree of each peer to be 12, 16 and 24 which results in $bwpf$ of 66, 100 and 134 Kbps, respectively. To satisfy the bandwidth-degree ratio for the above settings, the outgoing degree of high and low contributors are 18 and 3 ($bwpf$ of 66 Kbps), 24 and 4 ($bwpf$ of 100 Kbps) and finally 36 and 6 ($bwpf$ of 134 Kbps), respectively. The peer population is 200 and source bandwidth is equal to the stream bandwidth as 1.6 Mbps. The buffer size is computed as discussed above ($\log_{OutDeg}(N/Deg_{src})$+3) and set to 6 intervals (each interval is 6 sec in this set of simulation). The simulation is run for 6000 seconds and we model peer inter-arrival and session time based on prior empirical studies on deployed P2P streaming systems [129, 87]. Figure 61. shows the 10th, 50th and 90th percentile of average delivered quality among participating peers as a function of their incoming degree. Each line represents a different $bwpf$. This figure clearly demonstrates that for a fixed $bwpf$ the delivered quality to individual peers is directly proportional to their incoming degree, regardless of the parameter choice of $bwpf$. Moreover, by increasing $bwpf$ with a smaller incoming degree peers can receive higher quality. This result implies that the packet-level dynamics on content delivery has minimal impact on the delivered quality when a well-designed block scheduling algorithm and sufficient buffer size are deployed at each peer. Therefore, for a fixed $bwpf$ the incoming degree of each peer is a good estimator of its observed quality (or its allocated share of resources). [2]

6.3. Overview of Tax-based Resource Management

The primary goal of a contribution-aware scheme is to enable individual peers to determine their share of available resources (*i.e.*, bandwidth) in the system based on the amount of resources they contribute as well as the aggregate amount of available resources in the system. Given the direct relationship between the (incoming and outgoing) bandwidth and the (incoming and outgoing) degree due to the bandwidth-degree

[2] Note that, the choice of $bwpf$ is limited by the peer with minimum bandwidth contribution and it can be set to a fraction of it.

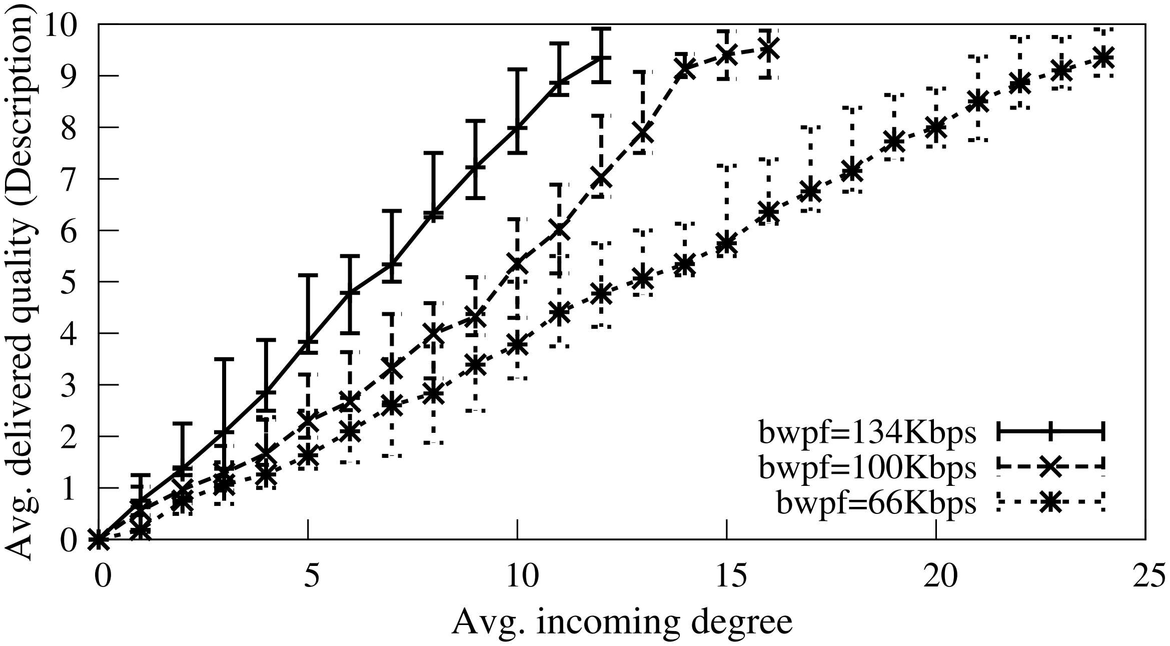


FIGURE 61.: Relationship between incoming degree and average delivered quality of individual peers.

constraint as described in Section 6.2., the contribution-aware scheme in the context of live mesh-based streaming can be formulated as deriving the incoming degree of a peer based on its outgoing degree. More specifically, the goal of each peer is to determine its incoming degree (*i.e.*, the number of parents) based on *(i)* its outgoing degree (*i.e.*, the number of child peers), and *(ii)* the aggregate outgoing degree across all peers.

To support the contribution awareness, each peer i uses a generic cost function[96] to determine its incoming degree R_i:

$$R_i = \frac{1}{t}W_i + \frac{t-1}{t}\sum_{j=1}^{N}\frac{W_j}{N}, \tag{VI.1}$$

where t, N, and W_i denote the tax rate in the system, number of participating peers, and the outgoing degree that peer i is willing to contribute. In essence, R_i presents the "entitled" share of system resources for peer i and thus we refer to R_i as *entitled* degree. As shown in Eqn. (VI.1), R_i is the sum of two terms. The first term represents the incoming degree of a peer due to its own contribution (W_i). The tax rate is always greater than or equal to one ($t \geq 1$) to balance supply and demand for resources in the system. The second term represents an even share of these extra resources among participating peers. This share of excess resources depends on the group state, namely group population (N) and the amount of aggregate available resources in the system ($\sum W_j$).

We assume that the tax rate t is a configuration parameter and thus known to each participating peer. If the group state information is known to individual peers, they can use Eqn. (VI.1) to determine their entitled incoming degree. In subsection 6.3.2., we describe a mechanism to collect the required group state information and to distribute them to participating peers.

In practice, the following two issues also contribute to the extra resources. First, when the aggregate incoming bandwidth of a peer reaches the maximum stream bandwidth, it does not require extra incoming degree. This implies that the incoming degree of peers is limited by $D_{max} = \frac{BW_{max}}{bwpf}$, where BW_{max} denotes stream bandwidth.

We call a peer as *saturated* when its entitled degree exceeds the maximum required degree, *i.e.*, $R_i > D_{max}$. Second, the entitled incoming degree of each peer (R_i) can only take integer values. In order to avoid over-estimating the amount of allocated resources to each peer, we always use the floor of the resulting value from Eqn. (VI.1). We can revise Eqn. (VI.1) to address these two issues as follows:

$$R_i = \Big\lfloor \min\{(\frac{1}{t}W_i + \frac{t-1}{t}\sum_{j-1}^{j}\frac{W_j}{N}), D_{max}\}\Big\rfloor. \tag{VI.2}$$

To effectively utilize the excess resources in the system, the unsaturated peers can further increase their incoming degree. These extra incoming connections are referred to as *excess degree* and denoted as e_i. In summary, the total actual incoming degree of each peer (a_i) consists of two components: $a_i = R_i + e_i \leq D_{max}$. Note that it is difficult to determine the amount of aggregate excess resources in the system, due to the random and dynamic nature of excess resources. In subsection 6.3.3., we describe how individual peers determine their excess incoming degree in a distributed fashion.

Once a peer computes its entitled degree (R_i), it intends to identify D_{max} parents in the system. Towards this end, first each peer learns about a subset of participating peers through a bootstrap node. Then, it progressively contacts them to discover their ability to serve as a parent. Each peer first establishes R_i entitled connections and then explores the feasibility of establishing some excess connections as we describe in subsection 6.3.3.. The contribution-aware scheme

Symbol	**Definition**
N	total number of peers in the system
W_i	the willingness of peer i, measured by degree,i.e, its bandwidth contribution to the overlay divided by bandwidth-per-flow, $bwpf$
a_i	actual number of incoming degree for peer i
f_i	actual contribution (outgoing degree) of peer i
R_i	computed entitled incoming degree of peer i
e_i	actual excess incoming degree of peer i
τ	period of update
D_{max}	maximum required degree to get full quality live stream

TABLE 8.: Notations used throughout this chapter.

should be able to gracefully cope with the inherent dynamics of peer participation, or churn. To achieve this goal, two issues should be addressed: *(i)* individual peers should periodically determine their entitled incoming degree, and adapt their overall incoming degree accordingly; *(ii)* each peer should implement a *preemption policy* to fairly manage the allocation of its outgoing degree among requesting child peers. In essence, the preemption policy ensures that the available resources in the system, are proportionally allocated across participating peers. We describe the preemption policy at each peer in subsection 6.3.4.. Table 8. summarizes the notations used throughout this chapter.

6.3.1. Goals & Assumptions

Goals & Assumptions: We make the following assumptions throughout this chapter: First, the incoming bandwidth of each peer is larger or equal to streaming bandwidth.

This implies that each peer tries to increase its overall incoming degree to its maximum value (*i.e.*, D_{max}). This is a reasonable assumption since the bandwidth of a video stream with an acceptable quality is around 400Kbps to 600Kbps which is less than the incoming access link bandwidth for most of the today's Internet users as indicated in earlier studies [133]. Second, we assume that peers are well-behave and provide correct information about the number of child peers they can support (W_i), *i.e.*, the amount of resources they are able and willing to contribute to the system. This simplifying assumption allows us to focus on the dynamics of the contribution-aware scheme rather than security side-effects of uncooperative peers [3].

6.3.2. State Allocation & Reporting

The state collection and reporting mechanism performs two tasks: *(i)* collecting the required information from individual peers and determining the group-level information such as N and $\sum W_i$; and *(ii)* reporting the group level information to all participating

[3] Assuming cooperative users is not unrealistic since one can use incentive schemes [134, 83, 87] to ensure contribution of resources or deploy a P2P streaming system in a closed setting (*e.g.*, within setup boxes) to achieve the same goal.

peers in the system. We consider a simple centralized approach for both state collection and reporting through a bootstrap node. When a peer joins the system, it contacts a well-known bootstrap node and provides its willingness to contribute (W_i). During a session, each peer sends a heart-beat message to the bootstrap node once every τ seconds and reports the value of its dynamic properties including its actual outgoing degree (f_i) and incoming degree (a_i) along with its entitled degree (R_i) and the list of its parents. The bootstrap node maintains the following information for each participating peers (W_i, f_i, a_i, R_i, *list of parents*) and updates this information after receiving each heart-beat message. Each peer also sends a BYE message to the bootstrap node right before its departure. If the bootstrap node does not receive a heart-beat message from a peer for 2*τ seconds, it assumes that the peer has departed and remove its record. In a nutshell, the bootstrap node has an updated state of individual peers and thus can easily determine the group-level state such as N and $\sum W_i$. Note that the state information at the bootstrap node may not be perfectly accurate since the state of each peer is likely to change between consecutive updates.

The bootstrap node reports the most recent group-level state to all participating peers once every τ seconds. When a peer receives a new report from the bootstrap node, it determines the number of its entitled connections (R_i) using Eqn. (VI.2). If the value of R_i is larger than its current incoming degree, it continues the discovery for more parents. In contrast, if its entitled incoming degree has dropped, it increases the value of e_i accordingly. Note that peers do not explicitly disconnect their incoming connections due to the drop of R_i, rather they consider a larger number of existing connections to be excess connections. The preemption policy at parent peers disconnects a proper number of these excess connections based on the overall demand for excess connections among peers. This passive strategy for disconnecting connections reduces dynamics in the system. τ is a configuration parameter that determines the tradeoff between the freshness of state information at the bootstrap node and the signaling overhead. More specifically, increasing the value of τ reduces the signaling overhead associated with state collection and reporting at the cost of lower accuracy for the state information at the bootstrap node. The default value of τ is 10 seconds.

6.3.3. Parent Discovery

The goal of the parent discovery mechanism is to enable each peer to locate the required number of parents to establish the desired number of incoming connections. Each peer always establishes R_i entitled connections and then explores possibility for establishing excess connections (if it requires any). Note that each peer does not label its individual incoming connections as an "entitle" or "excess" connection. Instead, a child peer only keeps track of its actual number of connections (a_i) and its entitled degree R_i that is periodically updated after each report from the bootstrap node. This is feasible in mesh-based P2P streaming mechanism, because all connections have the same value and thus the total number of connections determines the delivered quality not the identity of those connections[4].

To establish an entitled or excess connection, each peer first obtains the contact information for a subset of participating peers that are likely to be able to accommodate more child peers from the bootstrap node. Since the bootstrap node maintains the state of all participating peers (*i.e.*, potential parents), it can identify potential parents and report a list of random subset of them to a requesting peer. More specifically, the bootstrap node identifies a random subset of participating peers that have at least one empty slot or a child that can be preempted by the requesting peer. In essence, the bootstrap node implicitly coordinates the connections among peers. This in turn increases the probability of success during the parent discovery process. It is worth noting that despite this coordination, it is possible that a parent rejects a request due to a recent change in its status.

Given such a list of potential parents, each peer *sequentially* contacts peers in the list, provides its local state (*i.e.*, W_i, a_i and R_i)[5] and requests the contacted peer to serve as its parent. A contacted peer determines whether to accept or deny a parent request based on the *local preemption policy* as we describe in the following subsection. Once a child peer receives a response from a parent, it updates the number of its entitled and excess connections accordingly and provides its updated

[4] In contrast, the contribution aware scheme for tree-based P2P streaming [95] must specifically label each connection because each connection provides a particular description.

[5] All other states that a parent might need can be derived from these information.

information at its next heart-beat to the bootstrap node. Each peer continues to establish connection to more parents until its incoming degree reaches its maximum value (or D_{max}). If the list of potential parents is exhausted, the peer will contact the bootstrap node to obtain a new list. When peer i's request for connection is rejected by a potential parent, its reaction depends on its current state as follows:

- *Looking for more Entitled Connections (a_i<R_i)*: In this case, a child peer immediately sends a request to the next potential parent in the provided list by the bootstrap node. This rather aggressive approach to discovery is reasonable because there must be sufficient resources in the system, for each peer to reach its entitled incoming degree.
- *Looking for more Excess Connections (a_i>R_i)*: In this case, a rejected request is an indication of limited availability of excess resources in the system. Therefore, the rejected peer waits for an interval t_{wait}, called *wait interval*, before it contacts another parents to establish a connection. The wait interval is exponentially backoff with each rejected request for excess connections as follows [95]:

$$t_{wait} = t_{min} * K * (e_i + \beta^{ret}) \tag{VI.3}$$

where t_{min} is the minimum backoff time, K is a random number larger than 1, β is the backoff factor and ret is the number of consecutive failures. t_{min} is set to 5sec and β is 2. This approach for determining wait time adaptively adjusts the frequency of attempts for establishing excess connections by individual peers and thus the aggregate demand for excess connections without any explicit coordination among peers.

We note that state collection, reporting, and parent discovery can be performed in a distributed fashion (*e.g.*, [95]). For example, similar to the multiple-tree-based P2P streaming approach, a peer can traverse the mesh in a systematic fashion (starting from the source) and examine each peer to find a proper number of parents with desired type. In the similar fashion, peers' information can be collected and then propagated through the overlay. While this distributed approach does not require a central coordination point which might affect the scalability of the scheme, it can introduce a heavy signaling load to those participating peers that are located closer to source and add new dynamics (or introduce other side effects) that can affect the overall behavior of the contribution-aware scheme. We believe that a simple central approach, enables the bootstrap node to perform passive coordination and improve the efficiency of parent discovery. Moreover, it can properly represent a contribution aware scheme in mesh-based P2P streaming and can be used in practice as well.

6.3.4. Local Preemption Policy

The local preemption policy determines how a parent peer reacts to a request for connection from a child peer. If the current number of child peers for a parent peer is less than the degree that it is willing to contribute (W_i), then a request for connection is always accepted. However, if the outgoing degree of a parent peer is fully utilized, then a new child peer A can only replace (or preempt) an existing child peer B if providing a connection to child peer A has a higher priority. The relative priority of a connection to peers A and B is determined in different scenarios as follows:

- *En-Ex Policy*: If peer A is looking for entitled connection ($a_A{<}R_A$) and peer B already has some excess connections ($a_B{>}R_B$), then a request by A can always preempt an existing connection to peer B. This policy allows a new peer to easily reach its entitled incoming degree by preempting excess connections from other peers.

- *Ex-En Policy*: If peer A is looking for an excess connection ($a_A{>}R_A$) when peer B only has entitled connections ($a_B{\leq}R_B$), then a request by A can not preempt an existing connection from peer B.

- *En-En Policy*: if both peers only have entitled connections, then A can only preempts the connection from B if the normalized incoming degree of A is less than B, *i.e.*, the following condition is satisfied: $\frac{r_A}{W_A} < \frac{r_B}{W_B} - 1$. This condition basically ensures that all peers proportionally increase their entitled incoming degrees. Note that the equation incorporates a hysteresis effect to prevent oscillating preemption between two peers.

A , B	**Entitled**	**Excess**
Entitled	Yes if $\frac{r_A}{W_A} < \frac{r_B}{W_B} - 1$	Yes
Excess	No	Yes if $e_A < e_B - 1$

TABLE 9.: Local preemption policies used by each parent in determining if a new peer A can preempt an existing child peer B to use that slot as a child for this parent.

- *Ex-Ex Policy*: if peer A is looking for excess connections (a_A>R_A) and peer B has some excess connections (a_B>R_B), A can preempt an existing connection to peer B when it has a smaller number of excess connections (*i.e.*, e_A < e_B - 1). This condition balances out the number of excess connections among peers. It also incorporates a hysteresis to prevent oscillating preemption between two peers.

Table 9. summarizes the above preemption policies by a new peer A to an existing child peer B.

We illustrate how to use the rules in Table 9. through an example as follows: Suppose that peer b is already connected to a particular parent. In the first instance both peer A and peer B have entitled degree. The actual contribution (outgoing degree) of A is W_A. The actual entitled incoming degree of A is r_A. The actual excess incoming degree of A is e_A. Similarly for B. If $W_A = 20$, $r_A = 2$ and $e_A = 0$, then $(r_A + e_A)/W_A = 2/20 = 1/10$. If $W_B = 20$, $r_B = 5$ and $e_B = 0$, then $(r_B + e_B)/W_B = 5/20 = 1/4$. Since the calculation for $A < B$, A can preempt B. In the second instance, A has entitled degree and B has excess degree. Using the same values for the parameters of A, A once again has a calculated value for

$(r_A{+}e_A)/W_A = 2/20 = 1/10$. If $W_B = 5, r_B = 2 and e_B = 1$, then $(r_B{+}e_B)/W_B = 3/5$. Once again since the calculation for $A < B$, B can preempt A. In the third instance, A has excess degree and B is entitled. In this case A cannot preempt B. In the fourth instance, both A and B have excess degree. If $W_A = 5$, $r_A = 2$ and $e_A = 0$ then $e_A/W_A = 0/5 = 0$. If $W_B = 5$, $r_B = 2$ and $e_B = 2$, then $e_B/W_B = 2/5$ so A can preempt B since the ratio e_A/W_A is less than the ratio e_B/W_B.

Note that when a new peer joins the system or an existing peer loses its parent due to preemption, they start the parent discovery process and could in turn preempt another peer in the system. Therefore, the observed rate of change in parents among participating peers is higher than parent departure rate that occurs only due to churn. In essence, the preemption further aggravates the instability of the overlay.

6.4. Understanding Tax Function

Before evaluating the proposed contribution-aware scheme, we examine the behavior of the tax function (*i.e.*, Eqn. (VI.1)) as well as the impact of main parameters on its behavior (*e.g.*, W_i). Understanding the behavior of the tax function reveals how available resources are shared among participating peers across the parameter space in the absence of any dynamics in peer participation. This in turn serves as a reference to examine the performance of the contribution aware scheme and helps us examine the behavior of our proposed scheme over a proper portion of the parameter space.

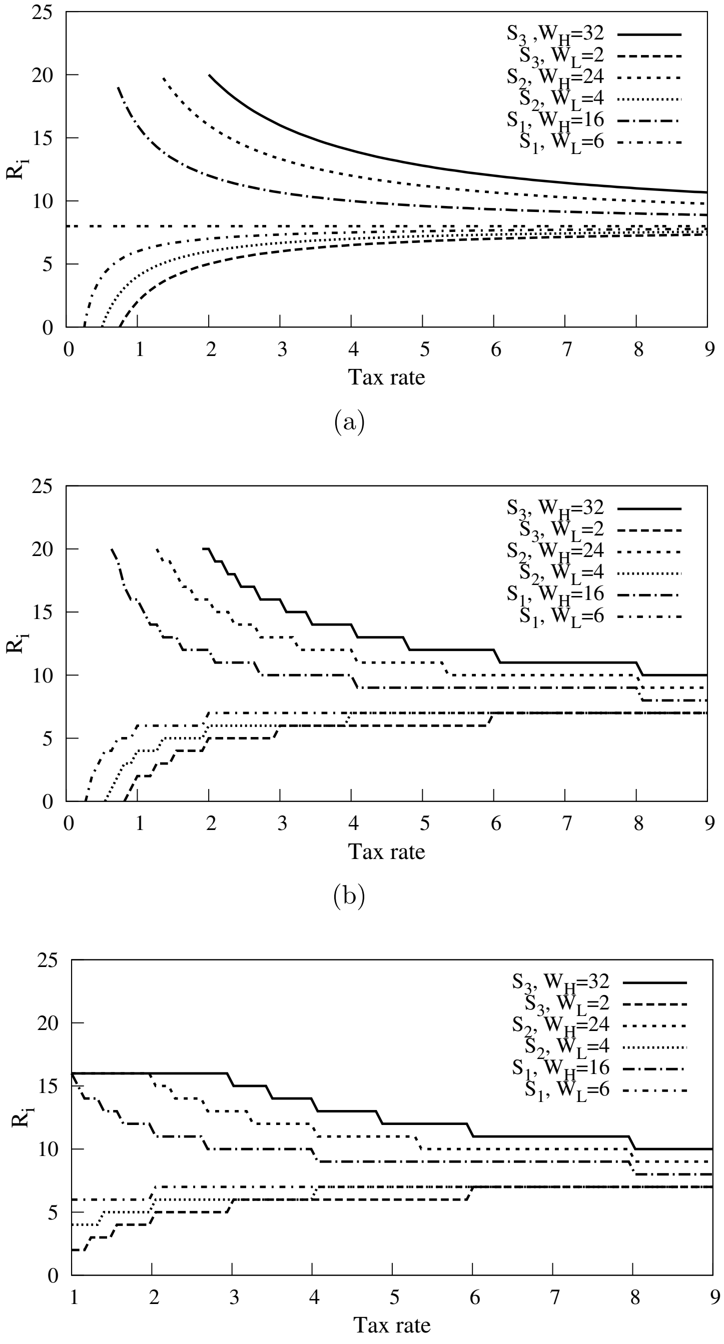
Given a scenario with N peers and their level of willingness to contribute (*i.e.*, outgoing degree W_i), we can define the Resource Index (RI) of a scenario as the ratio of aggregate contributed resource ($\sum W_i$) to the aggregate demand for resources. Since we assume that all peers have sufficient incoming bandwidth to receive full quality stream, the aggregate demand for resources can be simply determined as $N * D_{max}$ and thus RI is $RI = \frac{\sum W_i}{N * D_{max}}$. We can derive the value of $\sum W_i$, and replace it in Eqn. (VI.1) as follows:

$$R_i(t) = \frac{1}{t} W_i + \frac{t-1}{t} RI * D_{max}. \tag{VI.4}$$

Eqn. (VI.4) represents the entitled degree of a peer i as a function of tax rate t based on the following parameters: peer's willingness (W_i), resource index in the overlay (RI) and maximum incoming degree (D_{max}).

Figure 62.(a) plots $R_i(t)$ as a function of tax rate t for three different combinations of W_i when RI=0.5, $RI * D_{max}$=8 [6]. For comparison we plot a line for $RI * D_{max}$ in the figure. This figure reveals some important properties of the tax function across the parameter space as follows: First, as the tax rate increases, the entitled degree of high bandwidth peers ($W_i > RI * D_{max}$) is gradually decreasing with tax rate whereas for low bandwidth peers ($W_i < RI * D_{max}$) the entitled degree is gradually increasing. Furthermore, the entitled degree of all peers converges towards the same value of

[6] While this figure shows the tax function for positive tax rates values, in practice only tax values that are larger than 1, are of interest.



(a)

(b)

(c)

FIGURE 62.: Behavior of tax function with different values of W_i when RI is 0.5 and D_{max} is 16. (a) Basic tax rate function. (b) Floored tax rate function. (c) Floored tax rate function considering saturation.

$RI * D_{max}$ regardless of its initial value. To explain this, we note that as t increases the first term in the equation rapidly decreases and the second term converges to $RI * D_{max}$. Second, the larger the value of W_i, the faster the allocation of resources changes with tax rate. Third, the value of $RI * D_{max}$ approaches the value of the entitled degree of all peers when tax rate goes to infinite. Therefore, changing RI or D_{max} simply shifts the converging value in Figure 62.(a) up or down accordingly. Fourth, as we have discussed earlier, we always use the floor value of R_i to prevent over-estimating the available resources. Figure 62.(b) depicts $floor(R_i)$ (Eqn. (VI.4)) which results in a step-like evolution of entitled degree as a function of tax rate. Fifth, as we have explained earlier, high bandwidth peers become saturated when their entitled degree is larger than the maximum degree *i.e.*, $D_{max} \leq R_i$. This implies that the actual degree of a saturated peer is limited to D_{max}. Figure 62.(c) illustrates the upper limit of incoming degree for the saturated high bandwidth peers which occurs when the tax rate is low. Note that it is important to determine whether (and what fraction of) peers become saturated in a given scenario because this affects the amount of excess resources in the system which in turn determines delivered quality to non-saturated peers. We further elaborate this issue in the evaluation section.

In a nutshell, Figure 62.(c) represents the behavior of tax function in a static system where the peer population and the available resources are fixed and known, *i.e.*, the reference static scenario. In practice, because of the dynamics of peer participation and the resulting variations in available resources, the reported group state to individual peers is not perfectly accurate. Therefore, the average behavior among participating peers could be different from the above reference case. We investigate this issue in the next section.

6.5. Performance Evaluation

As we discussed in Section 6.2., in mesh-based P2P streaming mechanisms (such as PRIME) enforcing the bandwidth-degree ratio implies that all connections have roughly the same bandwidth. Furthermore, the swarming content delivery also implies that all connections have the same value. Therefore, main goals of the contribution aware scheme are *(i)* each peer has a proper number of child peers so that its resources are effectively utilized; and *(ii)* each peer can identify and establish connections with a proper number of parents proportional to its share of available resources. In essence, the performance of a contribution aware scheme for mesh-based P2P streaming should be assessed based on the ability of individual peers to keep their incoming and outgoing degrees at the proper values. Note that the delivered quality depends on both connectivity of the overlay that is managed by contribution aware scheme, and the swarming content delivery. However, as we have shown in Section 6.2., there is a direct relationship between peer incoming degree and quality. Therefore, to evaluate the performance of the proposed contribution aware scheme we only examine the connectivity among peers and do not consider the content delivery mechanism and the actual delivered quality.

Toward this end, we use our P2P session-level simulator, called *psim* which is introduced in Chapter IV [7]. *psim* is an event-driven simulator that incorporates pairwise network delay between participating peers using the King dataset[128]. Furthermore, *psim* incorporates a realistic model for churn by using a log-normal distribution (with μ=4.29 and σ=1.28) for peer session time and Pareto distribution (with a=2.52 and b=1.55) to model the peer inter-arrival time as reported by prior empirical studies on deployed P2P streaming systems [129, 87].

In our evaluations, we examine the impact of each one of the following factors on the performance of the tax-based contribution-aware mechanism for the live mesh-based P2P streaming approach: *(i)* Dynamics of parent selection, *(ii)* Benefits of Contribution-aware mechanism, *(iii)* Effect of Tax Rate & Peer Contribution, *(iv)* Resource Index (RI) in the system, *(v)* Scalability with group size, and *(vi)* Effect of update frequency.[8]

Each simulation is run for 6000 seconds and the information is collected during the steady state when the population reaches the desired target. The reported results for each simulation are averaged across multiple runs with a different random seed. We also use the following default parameters in our simulations: on average 80% of

[7] Note that, real world experiments and packet-level simulations are often useful to evaluate the protocol in a realistic setting such as realistic packet level dynamics (and background traffic), and bandwidth and RTT heterogeneity. However, we focus on session-level simulations as follows: the contribution-aware mechanism assumes all connections have the same value and primarily controls resource allocation by adjusting the incoming degree of the overlay. Therefore, this mechanism is not affected by packet level dynamics, bandwidth or RTT variations.

[8] One can compare the performance of tax-based contribution-awareness in both tree- and mesh-based approaches. However, due to the inherent differences between these two approaches[5], any observed differences in the performance of contribution-aware mechanism in tree and mesh-based will be related to major differences between them.

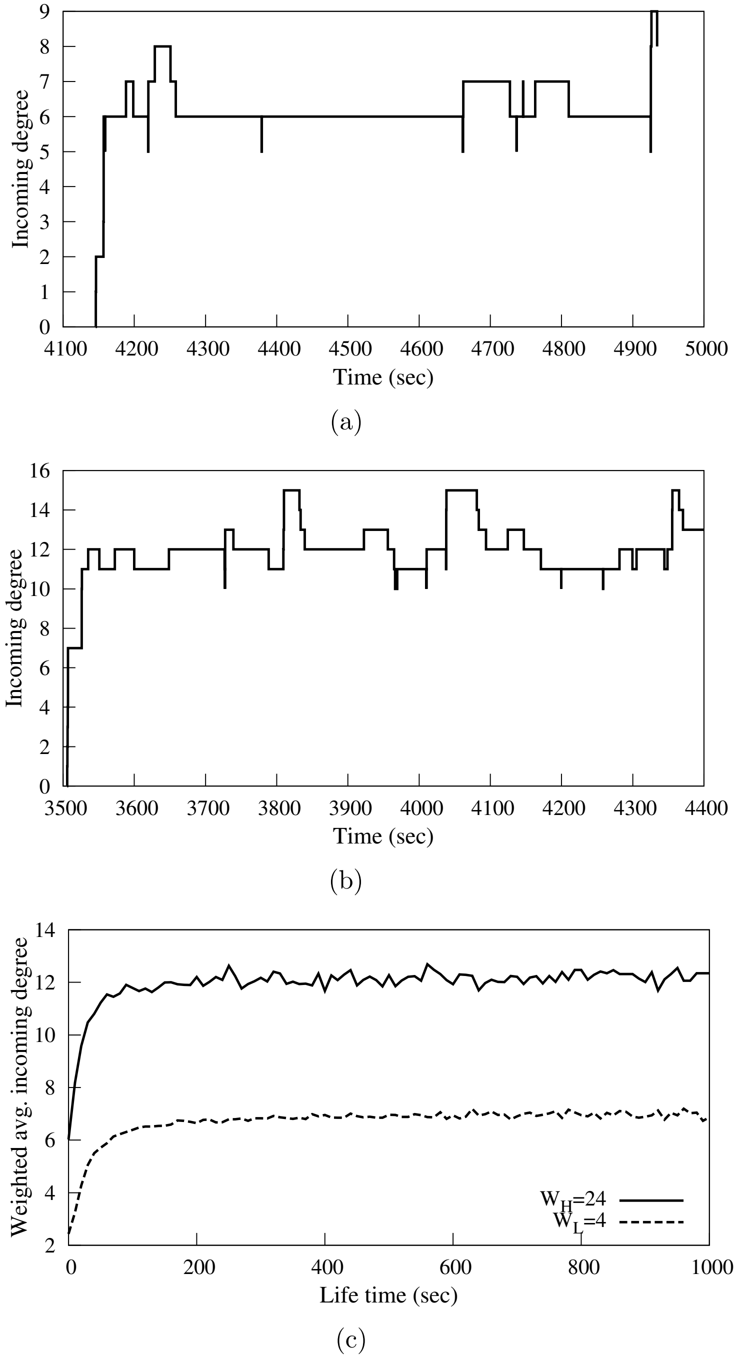


FIGURE 63.: Behavior of a high and low bandwidth peer ($t = 4$, $W_H = 24$ and $W_L = 4$). High and low bandwidth peers are entitled to degree of 11 and 4, respectively. (a) and (b) Variation of incoming degree for a low and high bandwidth peer, respectively. (c) Weighted average incoming degree of peers based on their life time.

peers are low bandwidth and the rest are high bandwidth, required incoming degree to receive full quality stream is 16, the degree of willingness for high and low bandwidth peers (*i.e.*, their outgoing degrees) are 24 and 4, respectively. The resource index is 0.5. The state collection and reporting is performed once every 10 seconds.

6.5.1. Dynamics of Parent Selection

We start by examining the dynamics of changes in the number of parents that are caused by the contribution aware scheme as well as churn. Figure 63.(a) and 63.(b) show the typical evolution of the incoming for a low and a high bandwidth peers over time when tax rate is 4, respectively. In this scenario, the average entitled degree for high bandwidth peers is 11 and for low bandwidth peers is 6. These figures illustrate that a peer can quickly increase its incoming degree from zero to reach its entitled degree, *i.e.*, less than 20 seconds for a high bandwidth peers and 11 seconds for a low bandwidth peer. These figures also show that once the incoming degree of a peer reaches its entitled degree, its incoming degree oscillates around the entitled value due to the minor changes in available resources and the variations in the number of excess connections. Figure 63.(c) presents the average incoming degree among peers whose lifetime is within the range of [x, x+10] seconds. In essence, this figure shows the evolution of average incoming degrees over time and reveals that all peers reach their target incoming degree in around 60 seconds. This also implies that peers with lifetime shorter than 60 seconds, will not remain in the system sufficiently long to reach their target degree.

6.5.2. Benefits of Contribution-Awareness

To examine the ability of the contribution aware scheme to manage the incoming degree of participating peers, we present the notion of "weighted average degree". Weighted average (incoming or outgoing) degree of a peer presents its effective average degree by weighting each degree by the interval that a peer maintained at that degree. For example, if a peer has an outgoing degree of 3 for one forth of its session and 5 for the rest of its session time, its weighted outgoing degree is 4.5. The weighted incoming and outgoing degree of each peer simply quantify the utilization and contribution of the resources during the session, respectively. We further divide the weighted average incoming degree of individual peers into weighted average entitled and excess degrees.

Figure 64. depicts the CDF of weighted average incoming degree among high and low bandwidth peers when tax rate is 2, with contribution-aware scheme (labeled as Cont.*) and without it (labeled as No-Cont.*). This figure clearly shows that in the absence of the contribution-aware scheme, the distribution of incoming degree is similar for high and low bandwidth peers, but it is rather diverse within each group, *i.e.*, the allocation of resources does not depend on the contribution of participating peers. In contrast, the distribution of incoming degree for high and low bandwidth peers are clearly separated and is very similar within each group. More specifically,

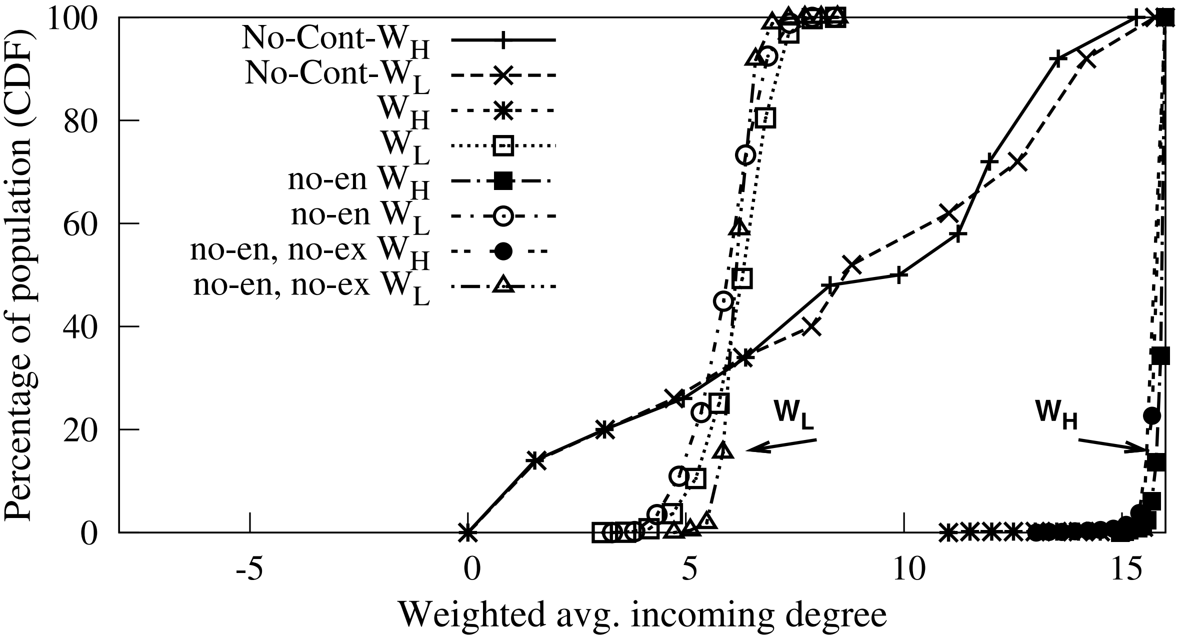


FIGURE 64.: Benefits of contribution-awareness: Distribution of weighted average incoming degree.

all low bandwidth peers (Cont.*WL) have a degree close to 7 whereas the degree of high bandwidth peers (Cont.*WH) is very close to 16. Figure 64. illustrates that the contribution aware scheme can effectively manage the allocation of resources among participating peers.

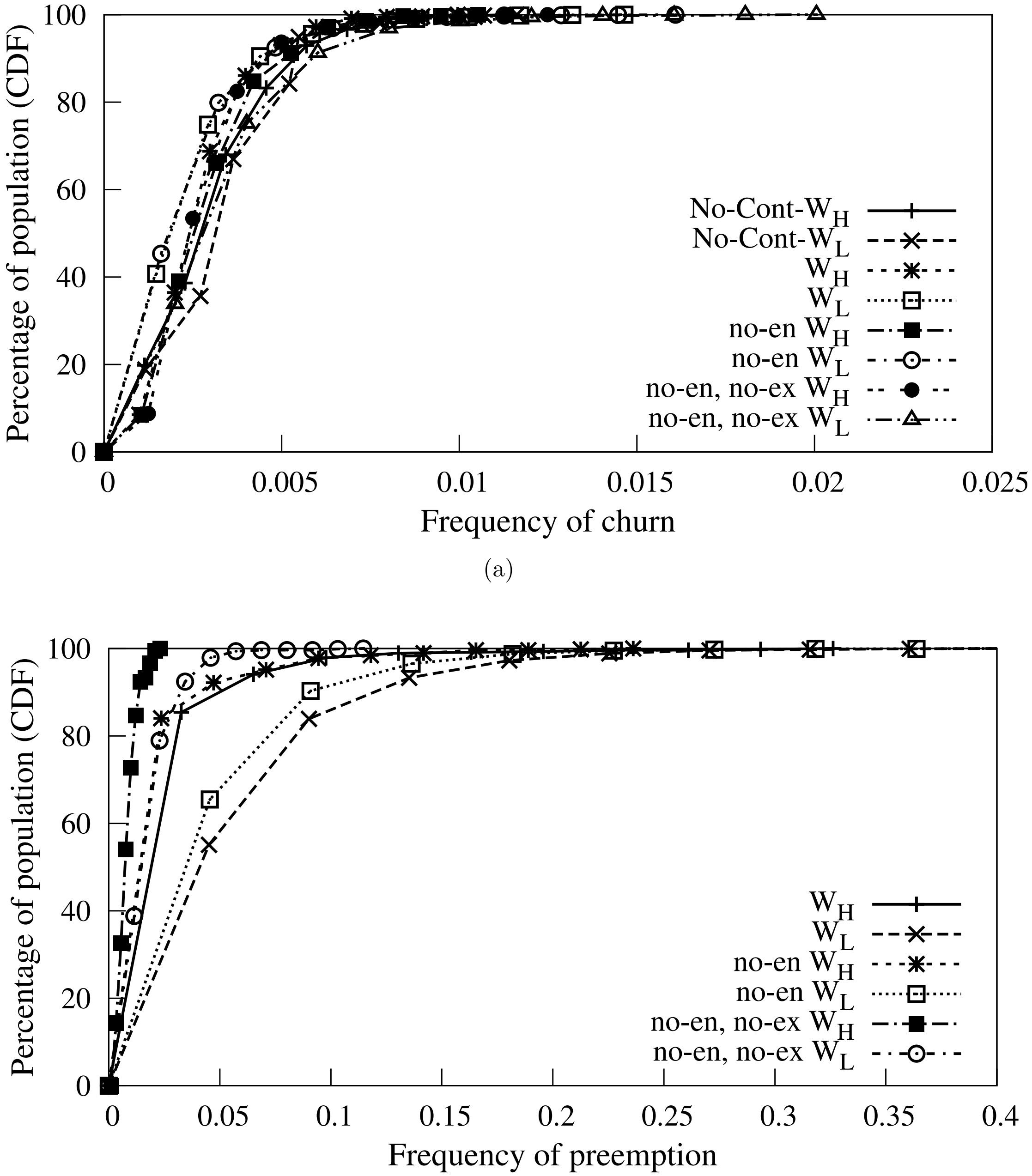
To quantify the importance of different preemption policies on the performance of the contribution aware scheme, we present the distribution of weighted average incoming degree for high and low bandwidth peers in two other scenarios where *(i)* the En-En policy is off (labeled as Cont-nop3-*); and *(ii)* both En-En and Ex-Ex policies are off (labeled as Cont-nop23-*). Figure 64. indicates that eliminating Ex-Ex and

En-En preemption policies does not lead to any visible change on the allocations of resources among peers. In other words, the En-Ex policy appears to be sufficient to achieve good performance.[9]

6.5.2.1. Stability of the Overlay

We also quantify the stability of the overlay by measuring the parent disconnection rates for individual peers. We further divide these disconnections into two groups: disconnections that are due to parent departure versus due to preemption by other child peers. Figure 65.(a) depicts the distribution of the average parent disconnection rate due to churn among both high and low bandwidth peers in all scenarios that we examined in Figure 64.. Since the overall parent disconnection rate for each peer due to churn is directly proportional to its incoming degree, we normalize the parent disconnection rate by the incoming degree in Figure 65.(a) for fair comparison. As expected, Figure 65.(a) illustrates that the normalized parent disconnection rate due to churn does not change with contribution aware scheme and does not depend on peer bandwidth (*i.e.*, peer degree). Figure 65.(b) presents the distribution of the average parent disconnection rate among participating peers for high and low peers only due to preemption in all the scenarios that we examined in Figure 64. (except for the scenario

[9] It is worth noting that En-En and Ex-Ex policies might affect the allocation of resources when RI significantly changes with time. However, constructing such a scenario requires detail information about potential dynamics of RI over time that has not been provided by previous empirical studies. We plan to further study this issue in our future work.



(a)

(b)

FIGURE 65.: Stability of the overlay: (a) Distribution of parent disconnection rate due to churn. (b) Distribution of parent disconnection rate due to preemption, with $t = 2$, $W_H = 24$ and $W_L = 4$.

Scenario	all changes	Churn	Preempt.
Cont.	1.5%	29%	2%
Cont. w/o En-En	3.2%	29%	5%
Cont. w/o Ex-Ex & En-En	24%	29%	51%
No-Cont.	29%	29%	100%

TABLE 10.: Percentage of stable peers for scenarios with different combinations of policies with and without contribution-aware scheme.

without contribution aware since no preemption occurs in that case) [10]. Figure 65.(b) shows that low bandwidth peers observe a higher rate of preemption in the base case (Cont.-WL) and even after disabling En-En preemption policy (Cont.-Nop3-WL). However, after disabling Ex-Ex and En-En, parent disconnection rate decreases significantly (Cont.-Nop23-WL). This suggests that the Ex-Ex preemption policy primarily contributes to the parent disconnection rate. Note that in this parameter setting high bandwidth peers' connections are entitled therefore they do not observe major preemption. We further examine stability in other settings in Subsection 6.5.3..

The stability of overlay can be also characterized in a more coarse-grained fashion. Table 10. presents the percentage of peers whose observed time between consecutive changes in parents (regardless of their cause) is at least 600 seconds. Each row of the table represents different scenario with contribution-aware scheme (including various combination of preemption policies) and without it. The table shows that in the absence of contribution-aware scheme 29% of peers are stable. The

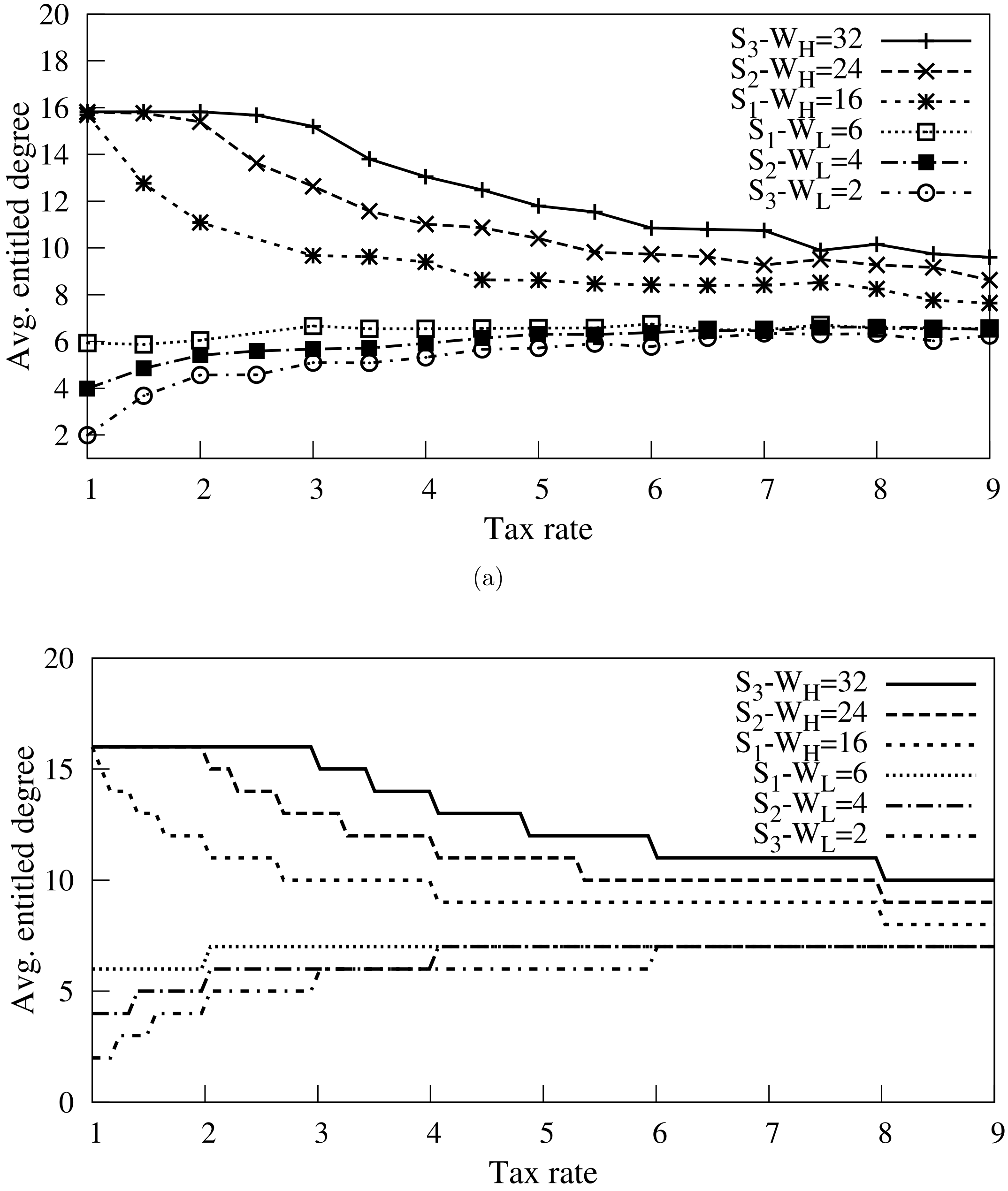
[10] Note that normalizing the rate of change in parents due to preemption in Figure 65.(b) is not meaningful since the observed rate depends on the relative number of excess connections for each peer.

percentage of stable peers with contribution aware scheme drops to 1.5%. Disabling the En-En policy slightly improves the percentage of stable peers from 1.5% to 3.2%. However, removing the Ex-Ex policy significantly increases the percentage of stable peers to 24% which is close the observed stability without the contribution aware scheme. Since the En-En and Ex-Ex policies significantly increase the instability of the overlay without affecting the performance of the contribution-aware scheme, we eliminate these two policies for the remaining evaluations in this chapter.

6.5.3. Tax Rate & Peer Contribution

In this section, we examine how the behavior of the contribution-aware scheme changes with the following two key parameters that determine a particular scenario: *(i)* the value of tax rate (t), and *(ii)* the value of peer's willingness to contribute (W_i). We consider the default parameters but with three different level of contribution (*i.e.*, degree of willingness or outgoing bandwidth) for high and low bandwidth peers as follows: *(i)* Scenario $S1$: WH= 16, WL= 6, *(ii)* Scenario $S2$: WH= 24, WL= 4 and *(iii)* Scenario $S3$: WH= 32, WL= 2.

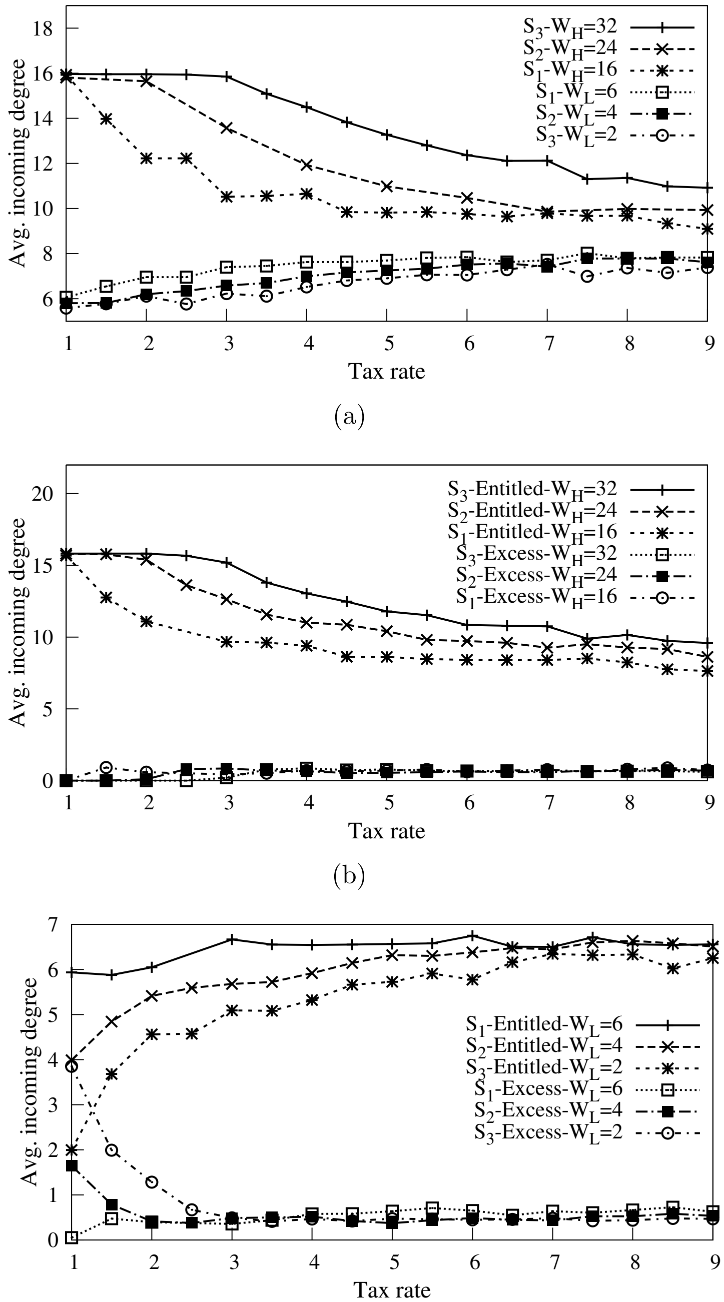
We want to keep the resource index (RI=0.5) and the percentage of high and low bandwidth peers (80% and 20%) fixed across these scenarios for proper comparisons. This implies that the heterogeneity of contributed resources by high and low bandwidth peers should proportionally adjusted across these scenarios so that the aggregate contributed resources remains fixed. Therefore, examining the



(a)

(b)

FIGURE 66.: Effect of tax rate: (a) Weighted average entitled degree. (b) Computed entitled degree.



(a)

(b)

(c)

FIGURE 67.: Effect of tax rate: (a) Weighted avg. incoming degree. (b) Average entitled and excess incoming degree for high bandwidth peers. (c) Average entitled and excess incoming degree for low bandwidth peers.

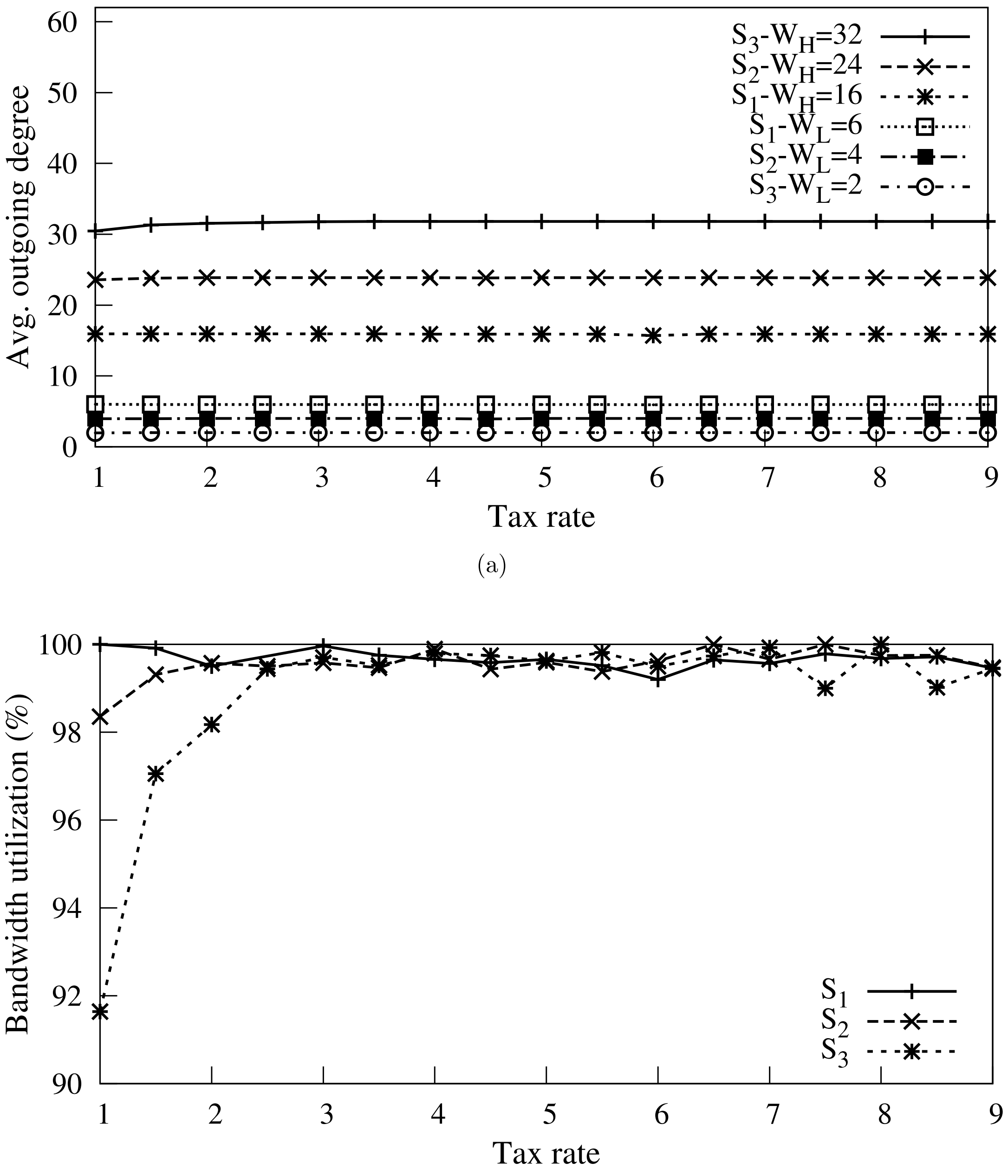
performance of the system across these scenarios reveals how the heterogeneity of contributed resources (or W_i) among peers affect system performance.

Figure 66.(a) depicts the weighted average entitled degree among high and low bandwidth peers as a function of tax rate for all three scenarios. Figure 66.(b) shows the entitled degree for high and low bandwidth peers based on Eqn. (VI.2) in all three scenarios as a reference. Comparing these two figures indicates that the weighted average entitled degree among high and low bandwidth peers closely follows its estimated values by equation (2) despite the existing dynamics in the connectivity among peers. Figure 67.(a) presents the weighted average of total incoming degree (both entitled and excess) among high and low bandwidth peers in three scenarios. This figure shows that except for very small tax values, the average values of entitled and total degrees are close.

To further examine the changes in entitle and excess degrees in each group of peers with tax rate, Figure 67.(b) depicts the weighted average value of both entitled and excess degree for high bandwidth peers in three scenarios whereas Figure 67.(c) presents the same information for low bandwidth peers. These two figures illustrate the following points: First, when tax rate is small, the entitled degree of the high bandwidth peers becomes saturated and thus they do not require excess connections. Since saturated peers do not use their entitled degree, excess resources becomes available in the system, and the amount of excess resources is proportional to (R_i - D_{max}), where R_i is the computed entitled degree of a high bandwidth peer i. Low bandwidth peers can utilize these excess resources as excess connections as shown in Figure 67.(c). The lower the entitled degree of low bandwidth peers is in these cases, the more available resources exist for excess connections. Thus low bandwidth peers can get larger number of excess connections as illustrated in Figure 67.(c). Second, as long as high bandwidth peers are not saturated (t>4), the average excess degree for both high and low bandwidth peers are the same and does not change with the tax rate or the distribution of peer contributions (across scenarios). The only reason for excess resources in these circumstances is the rounding of entitled degree (due to floor function). Since the amount of resulting excess resources does not change with tax rate or distribution of contribution by peers, the number of average excess degree remains fixed. This also shows that the contribution-aware scheme evenly divide excess resources among participating peers.

6.5.3.1. Utilization of Resources

To investigate the utilization of resources in the system, Figure 68.(a) depicts the weighted average outgoing degree among high and low bandwidth peers for three scenarios as a function of tax rate. This figure clearly shows that the outgoing degrees of peers in all scenarios are very close to their willingness to contribute (W_i), *i.e.*, the contribution-aware scheme can effectively utilize available resources for different distribution of resources among peers despite the dynamics of peer participation. Figure 68.(b) presents the overall utilization of outgoing degree among



(a)

(b)

FIGURE 68.: Effect of tax rate: (a) Weighted average outgoing degree. (b) Average bandwidth utilization.

all peers in one snapshot of the overlay. This figure shows that when high bandwidth peers are not saturated, resources are perfectly utilized. In the saturated region, the overall utilization of resources slightly drops due to the dynamics of excess connections. This is the reason for minor drop in the outgoing degree of high bandwidth peers for scenario S_3 in Figure 68.(a) when tax rate is small. To explain this, we note that a relatively larger fraction of resources in the system is utilized by excess connections in the saturated region. As the fraction of excess resources and thus excess connections increases, the probability of rejected request for an excess connection grows. This in turn reduces the utilization of resources due to the backoff in adapting the *waitinterval* for retrying a rejected excess connection request.

6.5.3.2. Stability of the Overlay

To quantify the stability of overlay, Figure 69. depicts the average parent disconnection rate due to preemption among high and low bandwidth peers across all three scenarios as a function of tax rate. Within the saturated region (t<4), high bandwidth peers do not experience any preemption simply because they only establish entitled connections that can not be preempted. However, outside the saturated region, high bandwidth peers experience a fair parent disconnection rate that gradually drops with increasing tax rate. The observed rate of disconnection by low bandwidth peers is small within the saturated region since there is not much contention for resources and thus no need for preemption. Outside the saturated

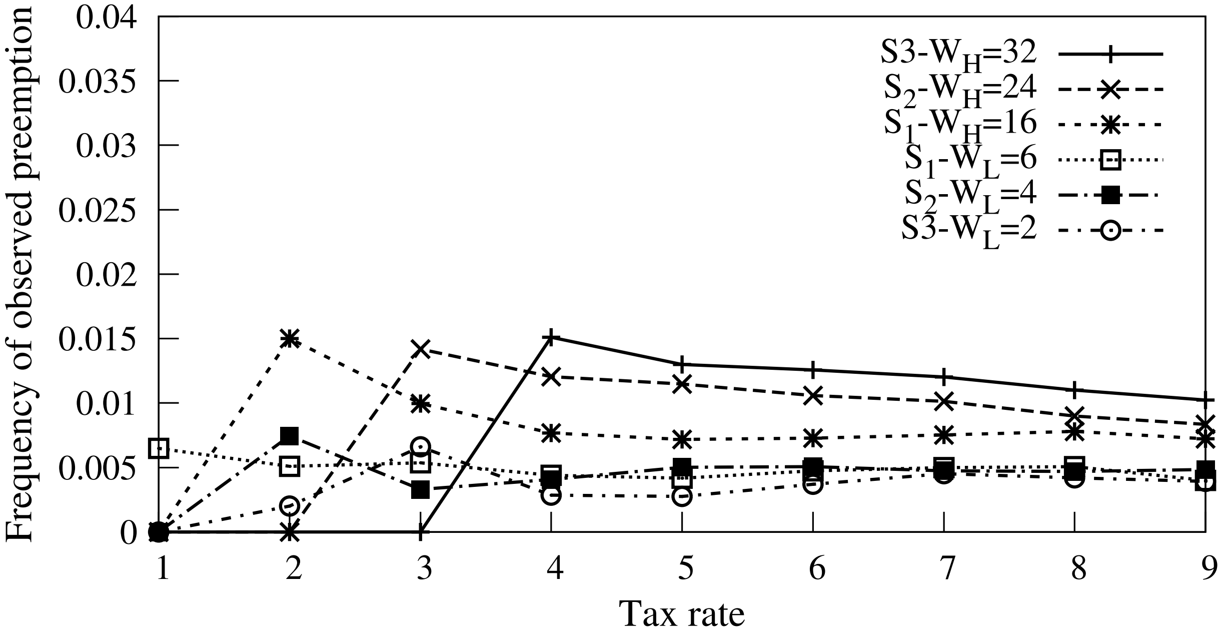


FIGURE 69.: Stability of the overlay: Median of parent disconnection rate due to preemption.

region, the average parent disconnection rate among low bandwidth peers does not change with tax rate across different scenarios. Moreover, while all participating peers have the same average number of excess connections outside of the saturated region, (as shown in Figures 67.(b) and 67.(c)), Figure 69. reveals that high bandwidth peers surprisingly observe a higher rate of disconnection.

The above trends in the stability of parent primarily depends on the average peer degree. More specifically, the larger the total peer degree, the higher the parent disconnection rate. To explain this issue, recall that the type of individual connections (*i.e.*, entitled vs excess) is not explicitly specified by the contribution-aware scheme in

Resource Index	BW Distribution	Contribution
0.5	12%-88%	40-4
0.8	23%-77%	40-4
0.9	29%-71%	40-4
1	34%-66%	40-4

TABLE 11.: Parameters used in simulations to examine the effect of RI.

the mesh-based P2P streaming, as we discussed in subsection 6.3.3.. Since each parent peer only uses the number of excess and entitled connections for its current children (based on their last update) in order to make preemption decisions, it is likely that two parents leverage their last update from their common child and simultaneously preempt (*i.e.*, disconnect) their connections to this child. The probability of such an event is proportional with the incoming degree of a child peer. Therefore, outside the saturated region, the change in stability as a function of tax rate is similar to the change in degree as shown in Figure 67.(a).

6.5.4. Resource Index

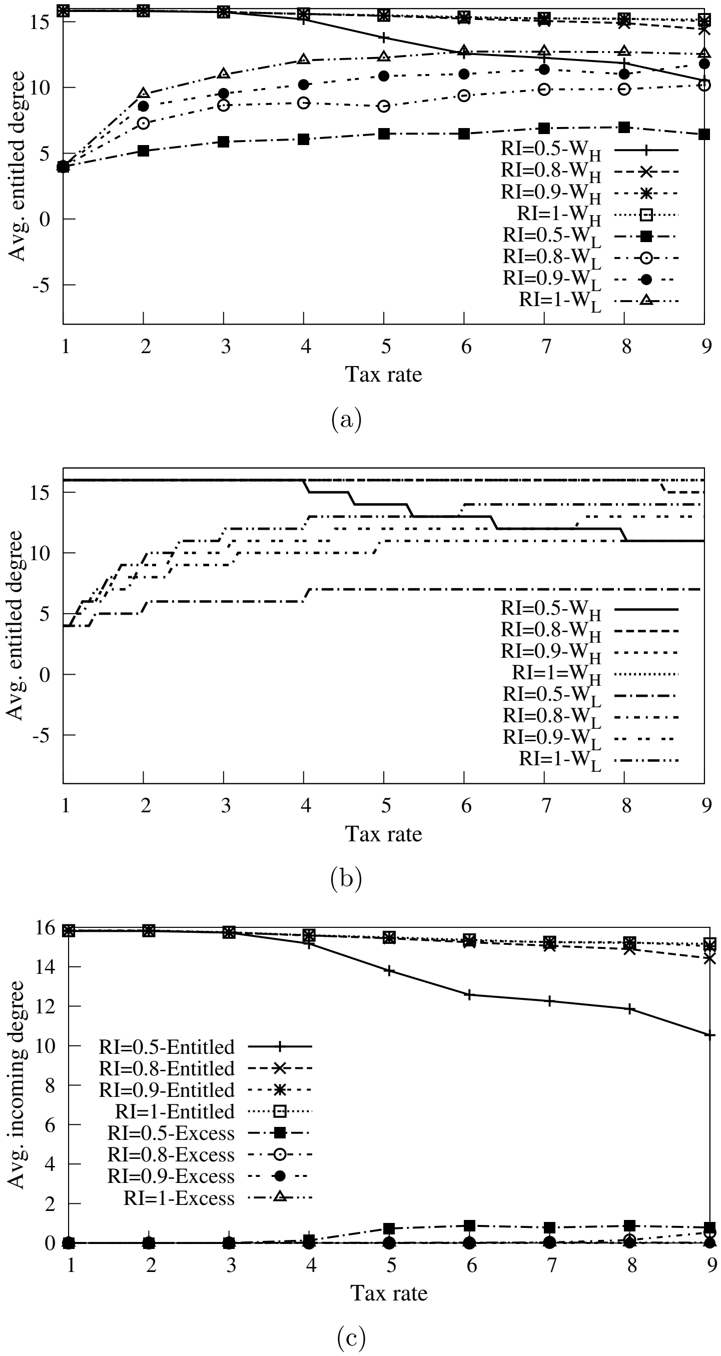
We examine the effect of resource availability (or RI) on the performance of contribution aware scheme. Toward this end, we keep the same level of heterogeneity for contributed resources where high and low bandwidth peers are willing to contribute 40 and 4 outgoing connections. However, we change the value of resource index by changing the percentage of high and low bandwidth peers as shown in Table 11..

Different scenarios in Table 11. are derived from reported traces by earlier empirical studies [95].

Figure 70.(a) depicts the weighted average entitled degree of high and low bandwidth peers as a function of tax rate for different scenarios. Figure 70.(b) shows the entitled degree of high and low bandwidth peers in the same scenarios based on equation (2) as a reference. Comparing these two figures reveals that the weighted average entitled degree of all peers generally follows their corresponding value derived from the equation. Figures 70.(b) and 70.(a) clearly illustrate that as more resources become available (*i.e.*, RI increases), high bandwidth peers remain saturated for a wider range of tax rates, *i.e.*, the size of the saturated region grows. The availability of extra resources enables low bandwidth peers to establish more excess connection and changes dynamics of the overlay.

To examine the effect of RI on each group of peers, we plot the average entitled and excess degrees for high and low bandwidth peers in Figure 70.(c) and 71.(a), respectively. Figure 70.(c) clearly illustrates the saturated region for high bandwidth peers in different scenarios where they do not have any excess connection. On the other hand, Figure 71.(a) reveals that low bandwidth peers manage to utilize the excess resources by establishing a larger number of excess connections within the saturated region for each scenario.

Figure 71.(b) shows the average out-degree of high and low bandwidth peers as a function of tax rate in scenarios with different RI. The figure clearly shows that

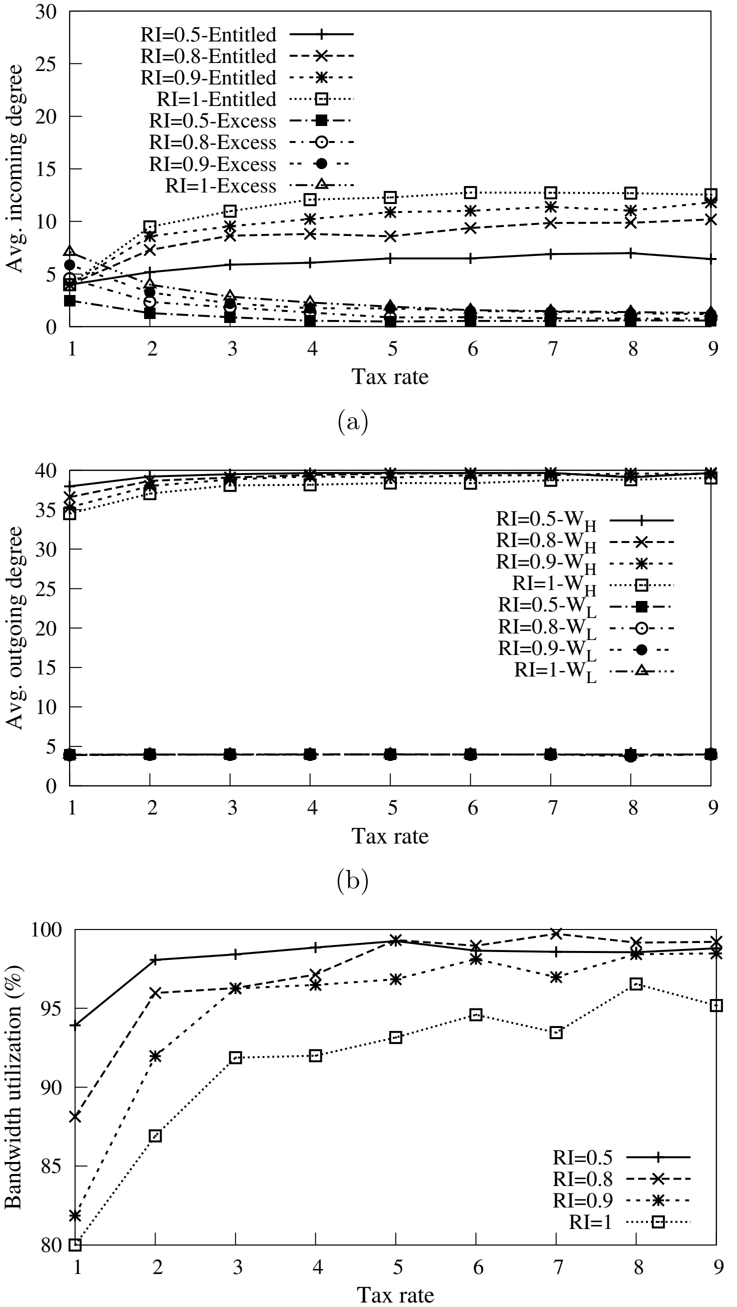


(a)

(b)

(c)

FIGURE 70.: Effect of resources: (a) Computed entitled rounded down degree. (b) Weighted average entitled degree. (c) Average entitled and excess incoming degree for high bandwidth peers.



(a)

(b)

(c)

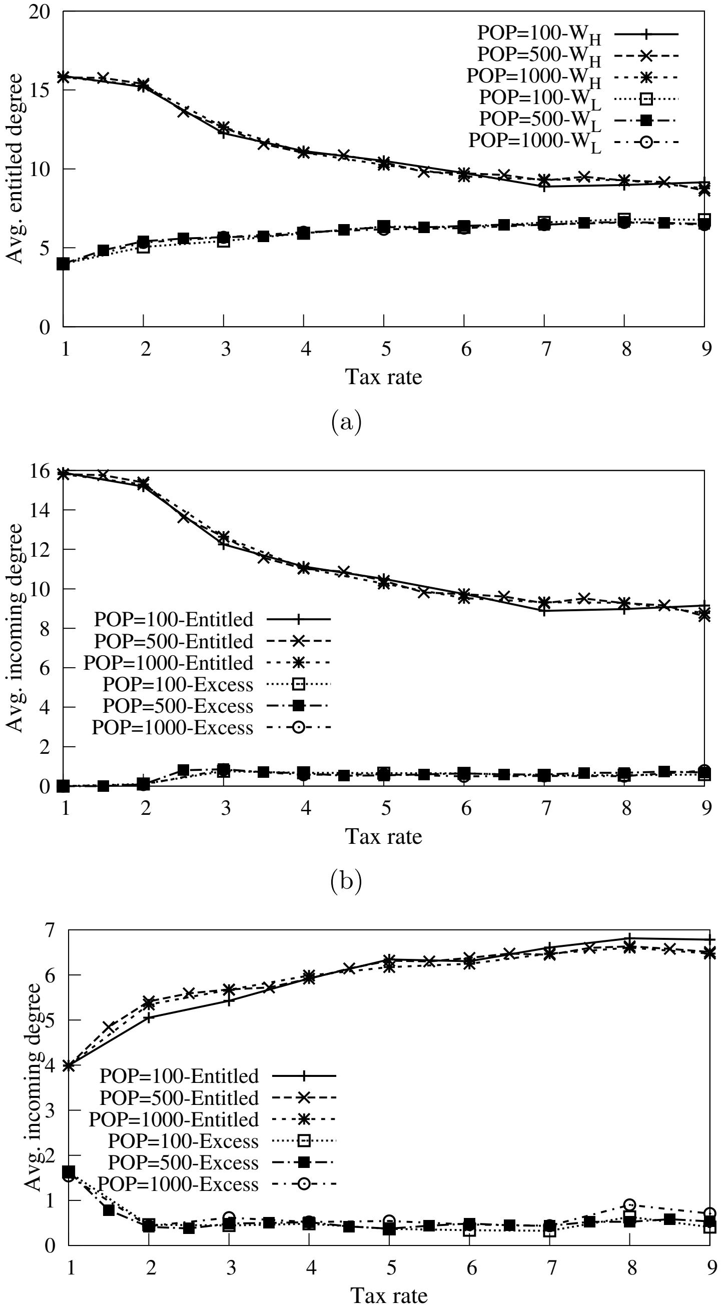
FIGURE 71.: Effect of resources: (a) Average entitled and excess incoming degree for low bandwidth peers. (b) Weighted average outgoing degree. (c) Average utilization.

across different tax rate and RI values, the average outgoing degree of high and low bandwidth peers is close to their maximum contribution. Figure 71.(c) presents the utilization of resources in a single snapshot of the system. This figure indicates that the overall utilization of resources is lower within the saturated region. The lower utilization of resources for both high and low bandwidth peer over small tax rate is due to the larger fraction of excess connections in these settings that results in a larger number of failed attempts to establish connection to a parent. This in turns lead to an exponentially increasing wait time which reduces resource utilization. We note that while exponential increase of *waitinterval* adjusts the aggregate demand for excess connection with the availability of resources, there is still a possibility of improper parent selection due to imperfect information on the location of available resources which leads to improper usage of resources. We have observed this effect in the subsection 6.5.3. over small tax rate as well.

6.5.5. Group Size

We now investigate how well the contribution aware scheme scales with the average number of concurrent peers in a session [11]. Toward this end, we change the average population from 100 to 1000 peers where $RI = 0.5$ and high and low bandwidth peers are willing to contribute up to 24 and 4 connections, respectively.

[11] Note that the total population changes with churn but psim can set the arrival rate in order to keep the average population at a desired number.



(a)

(b)

(c)

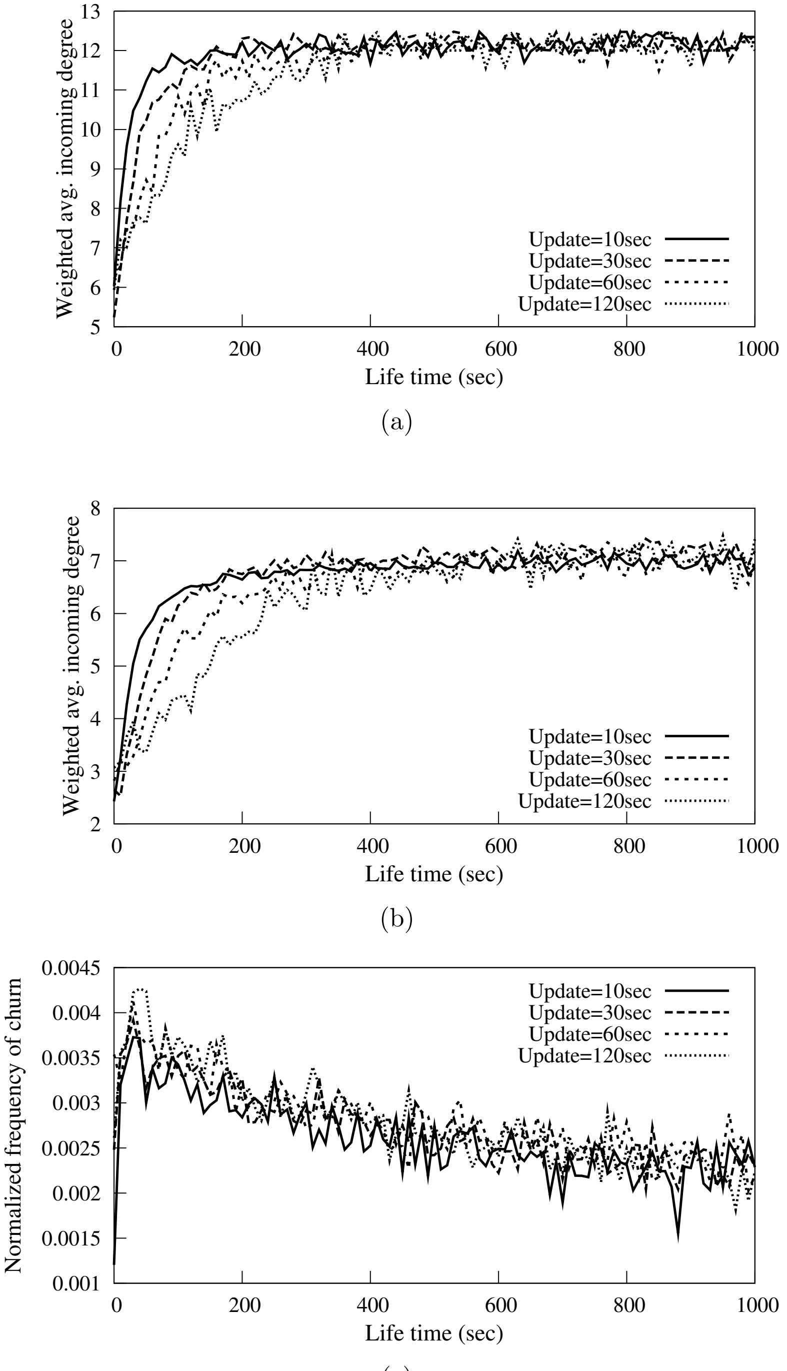
FIGURE 72.: Effect of peer population: (a) Weighted average incoming degree. (b) and (c) Average entitled and excess incoming degree for high and low bandwidth peers, respectively, as a function of tax rate.

Figure 72.(a) depicts the weighted average in-degree of high and low bandwidth peers as a function of tax rate for three different group sizes. Figures 72.(b) and 72.(c) show the average entitled and excess degrees of high and low bandwidth peers for different group sizes, respectively. These figures collectively illustrate that the average entitled and excess degree of low and high bandwidth peers are very close for different group sizes. This suggests that the contribution aware scheme is likely to scale with the number of participating peers.

6.5.6. Update Frequency

In this subsection, we explore the effect of reporting interval on the performance of the contribution aware scheme in a scenario where RI=0.5 and high and low bandwidth peers are willing to contribute up to 24 and 4 connections, respectively. In general as the update interval increases, the reported group state to individual peers and thus their estimate of available resources becomes obsolete. Underestimating the available resources will lead to a lower utilization of resources whereas overestimating could result in an imbalance allocation of resources in the absence of En-En preemption policy.

We first study the effect of update interval during the startup phase for individual peers when peers try to reach their target degree after arrival. Figure 73.(a) and 73.(b) depicts the average incoming degree among high and low bandwidth peers with lifetime between $[x, x + 10]$ seconds for different update intervals, respectively.



(a)

(b)

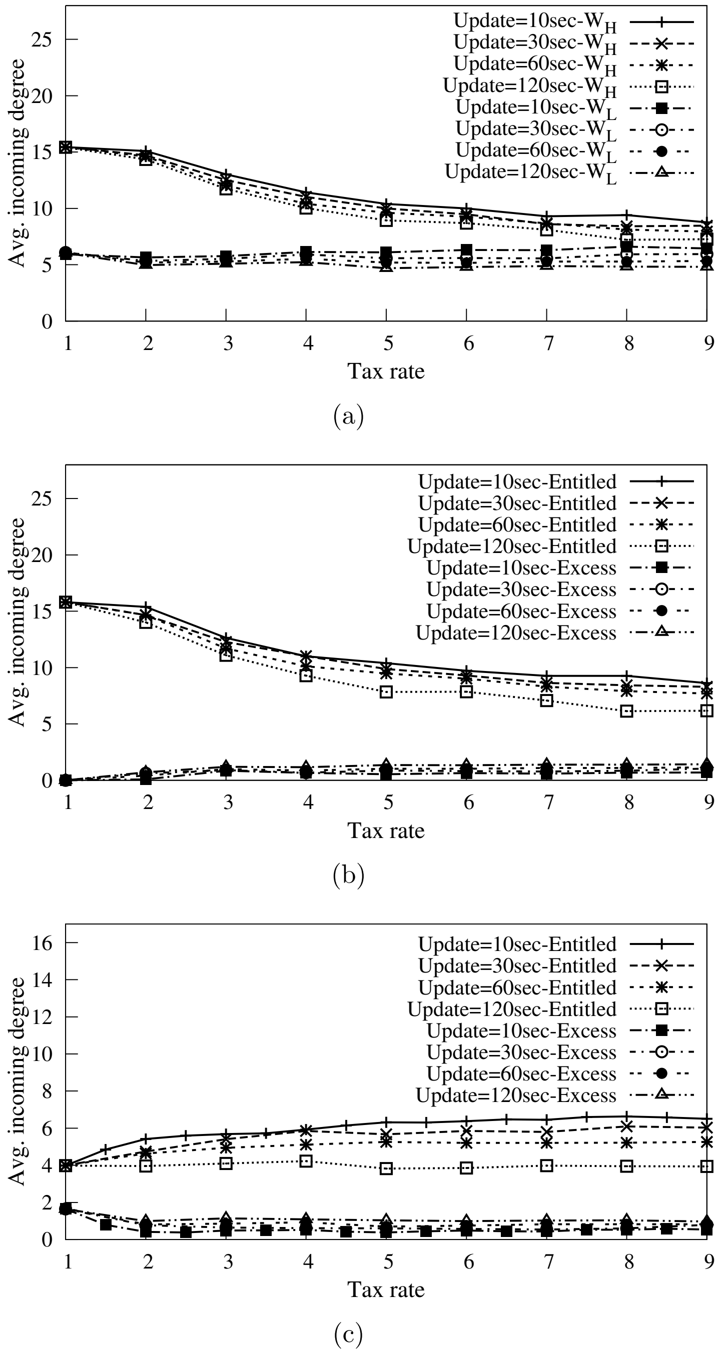
(c)

FIGURE 73.: Effect of update frequency: (a) and (b) Average incoming degree as a function of peer's life time for high and low bandwidth peers, respectively. (c) Normalized frequency of churn as a function of peer's life time.

The tax rate in these figures is 4 and the results for other tax rates exhibit similar behavior. We truncated the x-axis at 1000 seconds since the behavior remains the same for higher life time values. These figures clearly illustrate that increasing uptime primarily affects short-lived peers (with life time less than 400 seconds) that have not reached their target degrees. As the update interval increases, the effect is similar for both high and low bandwidth peers, and results in a lower incoming degree. To explain this result, we note that in our target scenarios, the group population and thus RI has a relatively small fluctuation due to churn[12].

Since the amount of aggregate resources is relatively stable, once long-lived peers establish their connections, the only change in their parents is due to churn. Therefore, increasing update interval does not have a major effect on them. However, short-lived peers are still building up their connections and are very sensitive to inaccurate information. Specifically, if a peer can not successfully identify all its entitled parents, it needs to wait until group state is updated at the bootstrap node to provide a proper list of parents. Inaccurate information could also affect ability of long lived peers to replace a departed parent. To quantify the frequency of such events, Figure 73.(c) depicts the average value of normalized frequency of churn among peers whose lifetime is between [x, x+10] seconds. This figure indicates that as peer's life increases, it observes the lower rate of churn among parents as well. This is simply

[12] One can generate artificial group dynamics that leads to significant and rapid changes in RI. However, such dynamics appear to be unrealistic since it is inconsistent with the reported peer arrival and peer session times in previous empirical studies.



(a)

(b)

(c)

FIGURE 74.: Effect of update frequency: (a) Weighted average incoming degree as a function of tax rate. (b) and (c) Average entitled and excess incoming degree for short lived high and low bandwidth peers, respectively, as a function of tax rate.

due to the fact that a connection between long-lived parent-child remains intact as long as aggregate resources do not change.

Figures 74.(b) shows the average entitled and excess incoming degree for high bandwidth peers that are short-lived (lifetime less than 400 seconds) for different update intervals. Figure 74.(c) depicts the same information for short-lived, low bandwidth peers. These figures illustrate a couple of points: *(i)* the overall trend of change in average degree with tax rate is similar for all update intervals; *(ii)* increasing the update interval results in a major drop in entitled degree and a minor increase in excess degree. These changes in the entitled and excess degrees are larger for higher tax rate. These trends can be explained as follows: as the update interval increases, it affects the ability of short-lived peers to quickly identify the desired number of parents due to the higher inaccuracy in the available group state at the bootstrap node. This leads to a lower utilization of resources and allows the excess connections to dynamically utilize a small fraction of this unused resource. Obsolete information affects only the second term in Eqn. (VI.2) ($i.e.$, $\sum_{i=1}^{N} f_i \leq \sum_{i=1}^{N} W_i$), by increasing tax rate, this term plays a more important role than the first term. This results in a larger difference in incoming degrees when update interval increases.

6.6. Summary & Future Work

This chapter presented a contribution-aware mechanism for live mesh-based P2P streaming based on the notion of tax function that depends on aggregate contribution of peers and the contribution of each individual peer. We examined the behavior of a commonly used tax function and described how it can be incorporated into mesh-based P2P streaming mechanisms to ensure proper allocation of resources among well behaved peers. We conducted extensive simulations to illustrate the ability of the proposed mechanism in proper allocation and high utilization of resources over a wide range of scenarios. Overall we have shown the effectiveness of our mechanism to allow peers receive different levels of quality based on the instantaneous available upload bandwidth in the system as well as the amount of their contribution.

Our main findings are summarized as follows:

- The behavior of the proposed contribution-aware scheme for mesh-based P2P streaming closely follows the theoretical model for allocated resources across different tax rates and resource indices.

- The performance of high bandwidth peers can be maximized when the value of tax rate is small. In fact, in our default simulation settings, the 10th, percentile of delivered quality to high bandwidth peers is improved by 800%.

- Increasing the tax rate has an opposite effect on the weighted average entitled and excess incoming degree for the high and low bandwidth peers.

- Comparing the effect of various preemption policies, reveals that some of the policies significantly increases the instability of the overlay. In our default simulation setting, the percentage of stable peers in a non-contribution-aware setting can be dropped by an order of magnitude by some of the preemption policies. In our proposed scheme, the percentage of stable peers reaches to the observed stability without the contribution-aware scheme.

- Increasing the intervals of reporting the group state, primarily affects short-lived peers in the overlay as they are more sensitive to the obsolete group state information. Moreover, increasing the reporting interval decreases the utilization of resources.

The work presented in this chapter, has a preliminary nature and we plan to pursue this work along the following directions: First, we would extend the notion of contribution awareness to a group of non-cooperative peers by enabling individual peers to securely report their own contribution to the system and reliably verify the contribution by other peers. Second, we plan to incorporate a pairwise incentive mechanism (similar to BitTorrent) between connected peers in a bi-directional overlay as an alternative approach and compare this approach with the tax-based contribution-aware approach presented in this chapter.

CHAPTER VII

OLIVES: OVERLAY-AWARE LIVE P2P STREAMING

Material in this chapter is adopted from a published [6] and one under submission work. The work is published in *IEEE MULTIMEDIA COMMUNICATIONS TECHNICAL COMMITTEE E-Letter* at October 2009, co-authored with Prof. Reza Rejaie. The other co-authors are Dr. Volker Hilt, Dr. Ivica Rimac and Prof. Markus Hofmann. The experimental work is entirely mine. The writing is primarily mine, with contributions by Prof. Reza Rejaie and Dr. Ivica Rimac. Co-authors provided technical guidance.

In P2P applications, participating peers often form an overlay which is largely agnostic to the underlying physical topology [109, 110]. This in turn increases the cost associated with P2P traffic for individual Internet Service Providers (ISPs) which is a serious concern. This problem has motivated the idea of localizing P2P traffic within individual stub ISPs by localizing the connectivity among their peers [111, 112]. The common assumption in this approach is that localizing the overlay connectivity has minimal impact on the performance of P2P applications. A few studies examined the performance of file swarming mechanisms over localized overlay and reported possible drop in performance in certain scenarios [114] or improvement due to potentially higher available bandwidth within an ISP [111].

Prior studies on this topic have primarily focused on the performance of file swarming mechanisms (*i.e.*, BitTorrent) over localized overlay. However, to our knowledge, the performance of live P2P streaming applications (*e.g.*, [64, 8]) over localized overlay have not been studied. As we have pointed out, compared to swarming content delivery, P2P streaming applications (especially for live streams) have more restricted timing requirements for delivery and more limited content availability due to the live nature of content. Given the growing popularity of P2P streaming applications in recent years and the volume of third associated traffic, incorporating the notion of "ISP friendliness" in P2P streaming application becomes increasingly important. In this chapter, we will present the design and evaluation of ISP-friendly P2P streaming mechanism for live video over the Internet.

7.1. Contributions & Design Objectives

This chapter investigates the design and evaluation of an ISP-friendly mesh-based P2P streaming mechanism for live content. To achieve ISP-friendliness in the context of mesh-based P2P streaming, we consider ”overlay localization”. Towards that, initially we investigate the maximum level of localization in overlay connectivity for live P2P streaming applications and the feasibility of achieving this goal in different scenarios. Further, we examine how the level of overlay localization affects the delivered stream quality by well-known block scheduling schemes, namely newest-first and random schedulings. Through analysis and simulations, we show that known block scheduling schemes can provide high quality stream only when individual ISPs have at least 200% redundant external traffic (3 times of stream bandwidth) and thus experience a large volume of external traffic. Furthermore, we identify fundamental underlying factors that limit the delivered quality to ISPs and peers as the overlay connectivity becomes more localized.

The main contribution of this chapter is the design and evaluation of a novel Overlay-aware LIVe P2P Streaming mechanism, called OLIVES, that maximizes the localization of streaming traffic within ISPs while providing a high quality stream to individual peers. In OLIVES, participating peers maintain a fully localized overlay within individual ISPs to effectively limit their external traffic. Peers adopt a two-tier overlay-aware block scheduling scheme to maximize their delivered quality in a fully localized overlay as follows:

- Inter-ISP scheduling that ensures the delivery of full quality stream to individual ISPs, and
- Intra-ISP scheduling that ensures the delivery of stream within each ISP.

7.2. Representing Swarming Content Delivery with Delivery Trees

In mesh-based P2P streaming, the collection of overlay connections that are used for the delivery of a block from source to all peers form a source-rooted spanning tree which is known as the *delivery tree* for a block. This notion of a *delivery tree* can be generally associated with a subset of blocks including a *substream* of a video. Without loss of generality, suppose a video source has D children and its bandwidth is equal to $STRBW$ which is sufficient to send only a single copy of individual blocks to the overlay ($D = \lceil \frac{STRBW}{bwpf} \rceil$). Then, all the delivered blocks to each child of source can be viewed as a *substream*. In the absence of any peer or bandwidth dynamics, if all connections have the same bandwidth, all blocks of a substream traverse the same delivery tree. In this simplified setting, the scheduling scheme at each peer in essence determines which substream to pull from each parent. Characteristics of delivery trees (for individual substreams) in this simplified setting demonstrate the basic performance of the scheduling scheme on a given overlay as follows: *(i)* the number of delivery trees that contain node n represents the delivered quality to this node, and *(ii)* the maximum depth d_{max} among all delivery trees indicates the required buffering at each peer as $buf(sec)$=d_{max}*Δ. Therefore, the goal of the scheduling scheme is to form D edge-disjoint spanning trees with minimum depth.

Properties of the delivery trees clearly depend on the interactions between the scheduling scheme and the overlay topology. We rely on the NR scheduling (newest-rarest) scheduling that has been introduced in Chapter III. Several other studies [127, 130] have shown that the NR scheduling scheme maximizes the diversity of blocks among individual peers and thus, results in the optimal delivered quality and delay for streaming of live video in realistic resource-constrained settings. Recall that, in this scheduling scheme, each peer requests the most recent blocks (with largest timestamp) from the corresponding parent. Requested blocks from other parents can be selected using a more-recent-first strategy. We assume that each block carries a hop count (OHC) for the number of visited peers. Therefore, each peer pulls new blocks with the smallest hop count from each parent (*i.e.*, shortest distance from the source)[1]. Note that, the hop count for block b at peer p indicates the depth of p on the corresponding delivery tree for b which also indicates the relative recency of the received block.

In Chapter III, we have shown that the *NR scheduling* [2] has two properties: *(i)* it outperforms other schemes in resource-constrained settings, and *(ii)* the maximum depth of delivery trees over a randomly connected overlay is limited to:

$$d_{max} \leq \log_D N + 1 + \frac{1}{1 - e^{-D}} < \log_D N + 3 \qquad \text{(VII.1)}$$

N and D denote the total number of peers and average peer (and source) outgoing degree, respectively.

[1] The rarest blocks among parents mostly have the minimum hop count.

[2] We note that different names have been used for this basic scheduling scheme in other studies. Furthermore, the same idea is used for tree construction in tree-based P2P streaming techniques [64, 21].

7.3. Achieving ISP-Friendliness: Design Space

The primary goal of incorporating ISP-friendliness into mesh-based P2P streaming mechanism for live video is to reduce the volume of external traffic for individual ISPs without compromising the delivered quality to the peers. This goal can be achieved by changing either the overlay connectivity or block scheduling or both. However, any such a design should consider the potential dependency between the overlay connectivity and block scheduling. Therefore, we identify broadly three alternative approaches to achieve ISP-friendliness as follows:

(i) **Revising Block Scheduling**: Given a random overlay, the block scheduling can primarily utilize the internal connections for pulling required blocks and use external connections in a demand-driven fashion [117, 118] when there is not an adequate number of new blocks among internal neighbors. Since new blocks must be initially pulled into each ISP through external connections, an adaptation scheme is needed to strike a balance between limiting the aggregate incoming external traffic and ensuring the delivery of high quality stream to the ISP. In this approach, the achieved reduction in external traffic is not deterministic but rather depends on the overlay topology and the adaptation scheme for using external connections.

(ii) **Revising Overlay Connectivity:** One can change the overlay connectivity to be more localized within each ISP for example by requiring each peer to establish a certain fraction of its connections locally [114]. This approach limits the volume of external traffic by reducing the number of external connections. However, the number of external connections and thus the amount of external traffic increases by the ISPs' peer population. An alternative approach is to limit the total number of external incoming (and outgoing) connections of an ISP and the associated traffic. As we show in the next section, increasing the localization of overlay connectivity beyond certain point decreases the ISPs delivered quality.

(iii) **Hybrid Approach**: One can change both block scheduling and overlay connectivity. While this seemingly offers many possibilities, the most promising one is to maintain a *fully localized* overlay and design an overlay-aware block scheduling scheme to ensure high delivered quality. *In a fully localized overlay, the number of incoming external connections are set to minimum such that their aggregate bandwidth is equal to the stream bandwidth.*

In this chapter, to achieve ISP-friendliness, we adopt the hybrid approach which by definition minimizes the external traffic of each ISP. The remaining challenge is to design a scheduling scheme that can deliver high quality stream over fully localized overlays. The scheduling scheme is crucial in order to mitigate the adverse impact that the localization may have as we discuss in the following section.

7.4. Overlay Localization: Maximum & Feasibility

The basic idea in external P2P traffic reduction for individual stub ISPs is to localize the connectivity of the overlay within each ISP. More specifically, enabling

Symbol	**Definition**
M_j	Population of peers in ISP_j
D	Incoming degree for each peer to receive the full stream ($\frac{STR_{BW}}{bwpf}$)
out_deg_i	Outgoing degree of peer i computed based on bw-degree constraint
C_j	Aggregate contribution of peers in ISP_j
C_{int_j}	Aggregate possible local contributions of peers in ISP_j
K_{in_j}	Number of incoming connections for ISP_j
K_{out_j}	Number of outgoing connections for ISP_j

TABLE 12.: Summary of used parameters.

each peer to connect to other peers within the same ISP reduces the number of external connections, thus, the volume of costly inter-ISP traffic.

7.4.1. Maximizing Overlay Localization for Live P2P Streaming

In the context of P2P live streaming applications, the aggregate incoming bandwidth to each ISP should be at least equal to the stream bandwidth to ensure that new blocks continuously “stream” to peers in that ISP. Assuming all overlay connections have roughly the same bandwidth ($bwpf$), maximum overlay localization is achieved when the number of incoming external connections for each ISP is set to its minimum value $D_{min} = \lceil \frac{STRBW}{bwpf} \rceil$, where $D_{in}(i)$ denotes the actual incoming degree of ISP i. Table 12. summarizes the parameters that we use through the chapter. Note that this requirement does not depend on the population of peers in an ISP. Given the minimum number of incoming external connections, we can define the redundancy in the actual incoming external connectivity of an ISP i as $R(i) {=} \frac{D_{in}(i) - D_{min}}{D_{in}(i)}$. Figure 75. shows a fully localized overlay with $D_{in} = 2$.

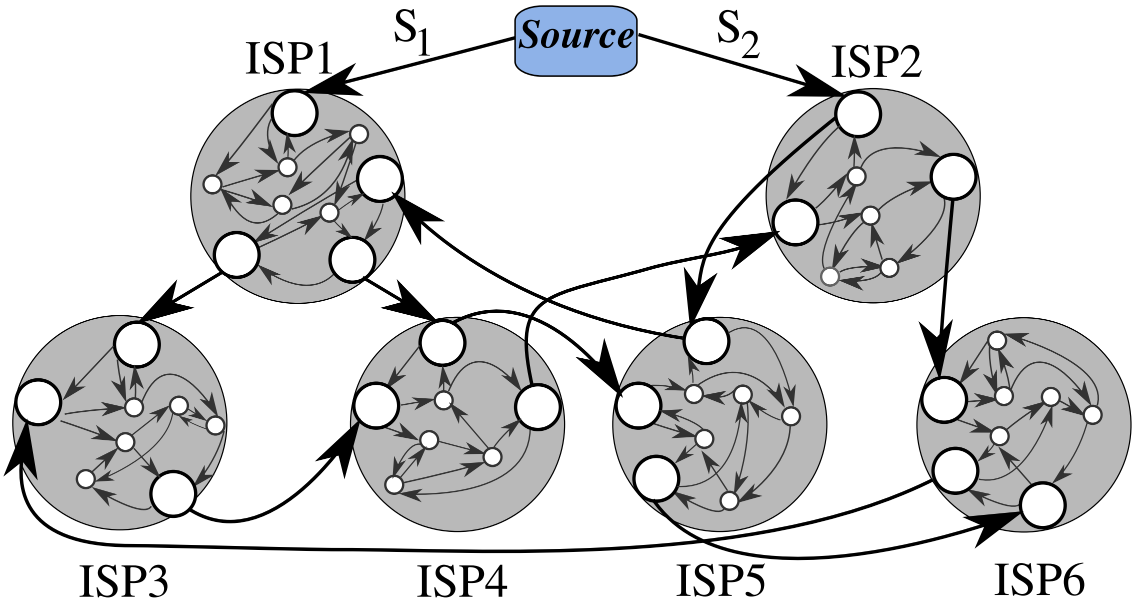


FIGURE 75.: A localized overlay with 6 ISPs while incoming and outgoing degree of each peer and external incoming and outgoing degrees of each ISP is 2.

7.4.2. Feasibility of Overlay Localization

Our goal is to investigate the feasibility of achieving maximum localization in connectivity for a given group of M peers in an ISP. Such a problem can be formulated as follows: a group of M nodes with certain in and out degree pairs $Deg{=}[PD_{in}(i), PD_{out}(i)]_{i=1}^{M}$ should be connected together such that

(i) there is at most a single edge (in each direction) between any pair of nodes, and

(ii) the number of un-established incoming and outgoing edges (*i.e.*, minimum external connections for ISP) are ID_{in} and ID_{out}, respectively.

If the number of unestablished edges are zero, the problem is essentially equal to determining whether an integer-pair sequence $P{=}[PD_{in}(i), PD_{out}(i)]_{i=1}^{M}$ is digraphic or there is a directed graph with degree sequence P which can be solved by Fulkerson theory [135].

Theorem 1- Let Deg=$[PD_{in}(i), PD_{out}(i)]_{i=1}^{M}$ be a negatively ordered integer-pair sequence. Then Deg is digraphic if and only if

$$\sum_{i=1}^{M}(PD_{in}(i)) = \sum_{i=1}^{M}(PD_{out}(i)) \tag{VII.2}$$

and for $s = 1, 2, ..., M$,

$$\sum_{i=1}^{s} min(PD_{in}(i), s-1) + \sum_{i=s+1}^{M} min(PD_{in}(i), s) \geq$$

$$\sum_{i=1}^{s}(PD_{out}(i)). \tag{VII.3}$$

To Allow ID_{in} and ID_{out} unestablished edges assuming Min=$min(ID_{in}, ID_{out})$ and Max=$max(ID_{in}, ID_{out})$ the above theorem can be modified by assuming Max additional integer-pair sequences in the forms of $[E_{in}(i), E_{out}(i)]_{i=1}^{Max}$ =

$$\begin{cases} (1,1) & \text{for } i = 1, ..., Min \\ \begin{cases} (0,1) & \text{if } ID_{in} > ID_{out} \\ (1,0), & \text{if } ID_{in} \leq ID_{out} \end{cases} & \text{for } i = Min, ..., Max \end{cases} \tag{VII.4}$$

Corollary 1- Let $EDeg^{-}$ be a negatively ordered integer pair sequence of integer pair sequence of $EDeg{=}P{\cup}K$, while $P{=}[PD_{in}(i), PD_{out}(i)]_{i=1}^{M}$ and $K{=}[E_{in}(i), E_{out}(i)]_{i=1}^{Max}$. If $EDeg^{-}$ is digraphic then a digraph over degree sequence P can be built considering ID_{in} and ID_{out} unestablished incoming and outgoing edges and given $L{=}\sum_{i=1}^{M}(PD_{out}(i))$, $ID_{out}{\leq}L$.

Proof:

(i) If all connections in set of K nodes are between P and K, then there should be ID_{out} connections from P to K and ID_{in} connections from K to P. Removing the Max number of peers in K, makes p a digraph with ID_{in} and ID_{out} unestablished incoming and outgoing edges.

(ii) If there is at least one connections from node k_1 to k_2 in K then

(a) there is at least one connection form p_1 to p_2 in P that can be cut to establish connections from p_1 to k_2 and k_1 to p_2 while **(b)** there is no existing connections between p_1 to k_2 and k_1 to p_2 previously.

To prove **(a)**, if there is no internal connection in P, then all connections from set of nodes in P are to the set of K. Assuming M is the number of those connections: *I. case* $M{=}ID_{out}$: then there should be least $M+1$ nodes in K in form of $(1,*)$ (to have one internal connection in K). The number of nodes in K in the form of $(1,*)$ is at most ID_{out} this means that all nodes in K in the form of $(1,*)$ have an incoming connection from set of nodes in P and there cannot be any internal connection in K, thus, P should have at least one internal connection.

II. case $M < ID_{out}$: then there should be least $M+1$ nodes in K in form of $(1,*)$ (to have one internal connection in K). The number of nodes in K in the form of $(1,*)$ is at most ID_{out} thus, M cannot be bigger than ID_{out} and case dismissed.

III. case $M{<}ID_{out}$: then as $M{=}L$, $L{<}ID_{out}$, which is against the problem assumption. Thus this case is also not valid.

(b) is also valid, as there cannot be any existing connection between p_1 to k_2 or k_1 to p_2 as incoming and outgoing of nodes in K are at most 1, so they cannot be connected to any other node previously and also a connection to each other.

In a setting where all peers and the ISP have the same in and out degree (*i.e.*, $PD_{in}(i){=}PD_{out}(i){=}ID_{in}{=}ID_{out}{=}D$), the basic requirement for the feasibility of localization is that peer population needs to be larger or equal to D. In the remainder of this chapter, we primarily focus on scenarios where maximum overlay localization is feasible by assuming an adequately large population of peers in individual ISPs.

7.5. Effect of Overlay Localization on Mesh-based Live P2P Streaming

One of the key question in swarming live video over localized overlays is *whether and how localization of the overlay connections affects the performance of content delivery for live streams?* For the discussion, we introduce the following differentiation amongst peers in an ISP:

- *edge peers* have at least one external incoming connection,
- *internal peers* do not establish any external incoming connections.

To analytically examine the effect of localization on the delivered quality, we consider an overlay with N peers in a *resource-constrained* setting. Suppose all overlay connections have the same bandwidth and there is no churn among peers. Then, all the blocks that the source delivers to its particular child experience the same delivery tree because of the deterministic nature of NR scheduling. We refer to a group of blocks that are delivered through the same delivery tree as a *substream*. In this simplified case, NR scheduling can be used at the granularity of *substreams* (instead of blocks) by pulling a substream with minimum overall hop count (OHC) from each parent. Given D distinct substreams, our goal is to derive the expected number of distinct substreams that reach ISPs (*i.e.*, delivered quality to each ISP) as a function of its number of incoming external connections (K) (*i.e.*, level of locality in its connections). This problem can be casted into determining the probability that given K samples from a basket of N balls that are equally divided into D distinct colors, at least one ball from each color is sampled.

The probability of not selecting a subtree of type t within K samples is:

$$
\begin{aligned}
P(n(t)) &= P(n(p_0) \cap n(p_1) \cap n(p_2) ... \cap n(p_{N_t})) \\
P(n(p_0)) &= 1 - \frac{K}{N} \\
P(n(p_0) \cap n(p_1)) &= (1 - \frac{K}{N}) * (1 - \frac{K}{N-1})
\end{aligned}
$$

Thus we have:

$$
\begin{aligned}
P(n(t)) &= (1 - \frac{K}{N}) * (1 - \frac{K}{N-1}) * (1 - \frac{K}{N-2}) \\
&\quad * ... * (1 - \frac{K}{N-(N_t-1)}) \\
&= \prod_{j=0}^{N_t-1} (1 - \frac{K}{N-j}))
\end{aligned}
$$

The expected value of K for having D distinct samples can computed by first introducing an auxiliary function as [136]:

$$
\delta(t) = \begin{cases} 0 & \text{If tree } t \text{ is not selected} \\ 1 & \text{If tree } t \text{ is selected} \end{cases} \tag{VII.5}
$$

The expected number of distinct trees is:

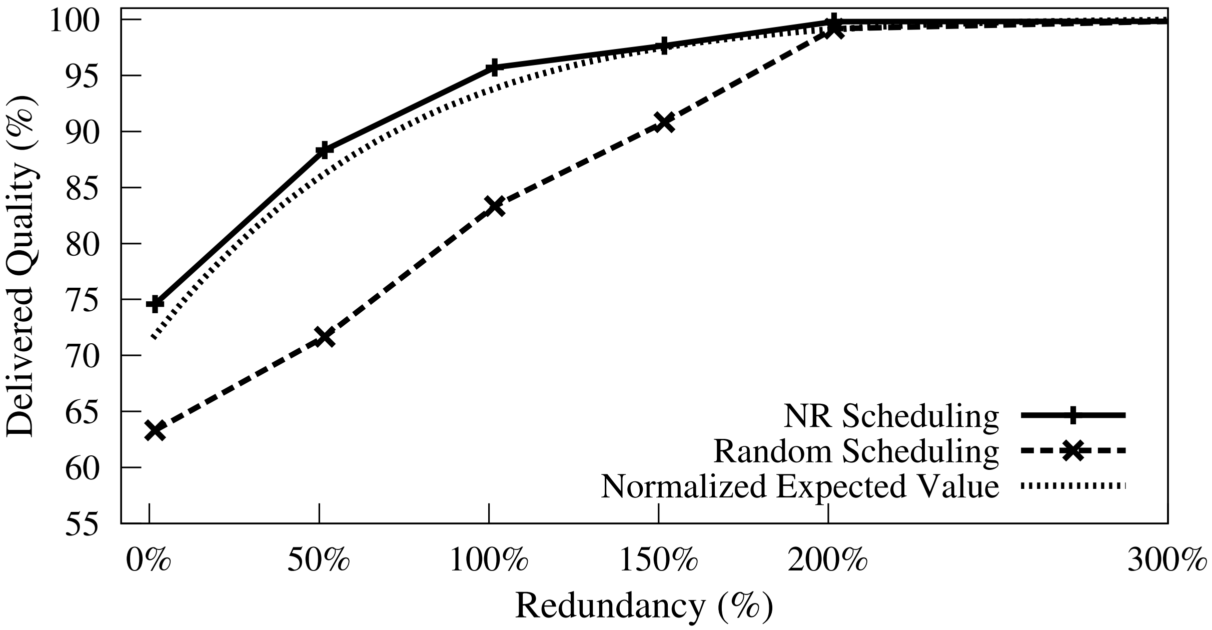
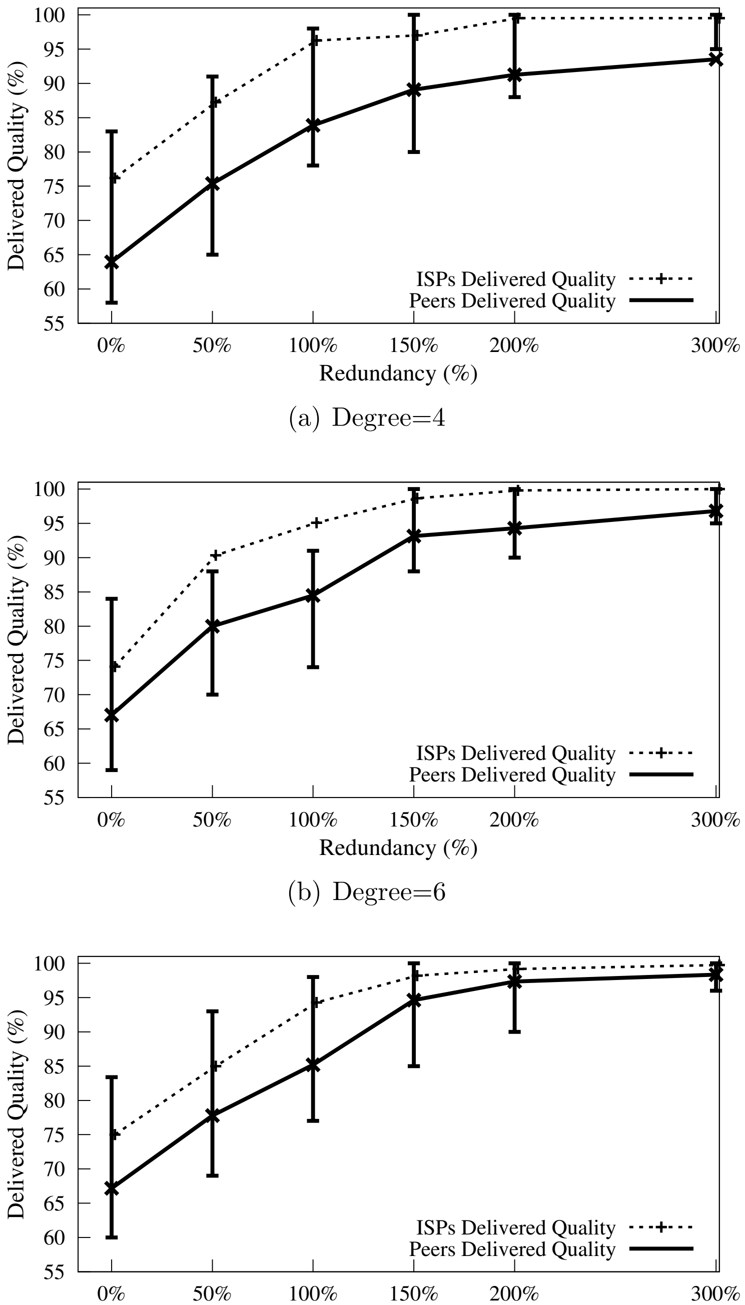


FIGURE 76.: ISPs delivered quality with NR scheduling and random scheduling, along with the expected distinct substreams divided by the number of substreams.

$$\begin{aligned}
E[Distinct] &= E\left[\sum_{t=0}^{D-1}\delta(t)\right] \\
&= \sum_{t=0}^{D-1} E\left[\delta(t)\right] \\
&= \sum_{t=0}^{D-1}(1 * P(\delta(t)=1) + 0 * P(\delta(t)=0)) \\
&= \sum_{t=0}^{D-1}(P(\delta(t)=1)) = \sum_{t=0}^{D-1}(1 - P(n(t))) \\
&= \sum_{t=0}^{D-1}(1 - \prod_{j=0}^{N_t-1}(1 - \frac{K}{N-j})) \qquad \text{(VII.6)}
\end{aligned}$$

To validate the above analysis, we simulate NP scheduling over an overlay with 5000 peers that are evenly distributed over 40 ISPs, whereby the incoming and outgoing degree of all peers are 12. We focus on a resource-constrained setting where the source sends a single copy of each block of video. While this simulation scenario captures only the basic behavior, it is suitable to reveal the major performance bottlenecks.

Figure 76. depicts the expected value of the number of distinct substreams normalized by the number of substreams (*i.e.*, $\frac{E[K]}{D}$) as well as the median ISPs' delivered quality in our simulations (with both NR and random scheduling) as a function of the redundancy in external connectivity of individual ISPs. This figure illustrates the following important points: *(i)* NR scheduling over a fully-localized overlay reduces the delivered quality by 25%. *(ii)* As redundancy in external connectivity of ISPs increases (*i.e.*, the overlay localization decreases), the overall performance gradually improves. To deliver a good quality to a large fraction of peers, at least 200% redundancy (3 times stream bandwidth) is required. *(iii)* Random scheduling results in lower ISPs' delivered quality than the NR scheduling in highly localized overlays. *(iv)* For large peer population, the effect of redundancy on ISPs' delivered quality by NR scheduling does not vary with the degree or peer population (as Eqn. VII.6 indicates), thus it exhibits the fundamental performance bottleneck caused by overlay localization as shown in Figures 77..



(a) Degree=4

(b) Degree=6

(c) Degree=14

FIGURE 77.: Peers and ISPs delivered quality with NR scheduling as a function of redundancy for various peer degrees.

7.5.1. Fundamental Performance Bottlenecks

Closer examination of our simulation results reveals the following two fundamental performance bottlenecks of NR scheduling over a localized overlay:

- **Misallocation of External Connections:** NR scheduling may result in improper mapping of external connections to substreams which in turn leads to the first two bottlenecks: *(i)* It may limit the delivery of substreams to only a subset of ISPs. Consider the overlay in Figure 78. for the delivery of two substreams S_1 and S_2 to a group of peers. Given the overall hop count (OHC) of both substreams at edge peers A and B, both edge peers C in ISP_3, and D in ISP_4, pull S_2 through their incoming external connections from ISP_1. This means that the delivery tree for S_1 is terminated at ISP_1 and this substream cannot reach other ISPs. *(ii)* Since incoming edge peers within each ISP independently determine the pulled substream from their external parents, it is likely that all the substreams are not collectively pulled into the ISP by all incoming edge peers.

  In Figure 76., the line labeled by "ISPs Delivered Quality" shows the median delivered quality to individual ISPs in our simulations. The low delivered quality to individual ISPs is due to a combination of the above two problems.

- **2) Misallocation of Internal Connections:** Even if all substreams are delivered to an ISP, NR scheduling may not result in a proper propagation of a

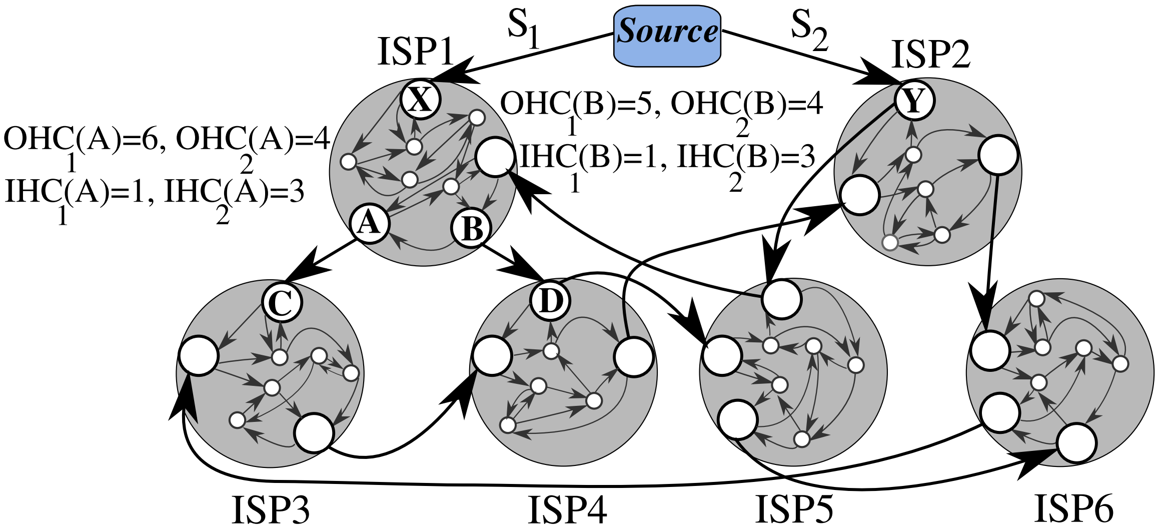


FIGURE 78.: A localized overlay with associated hop counts for two substreams of S_1 and S_2.

substream from an edge peer to all internal peers within an ISP. To demonstrate this problem, consider an ISP in Figure 79. whose edge peers A and B pull substream 1 and 2, respectively. Furthermore, peer A pulls substream 2 from an internal peer (peer B in this case) such that $OHC_2(A) = 3 < OHC_1(A) = 10$. As a result, internal peers C and D pull substream 2 from peer A due to a smaller hop count, which in turn makes substream 1 unavailable for other peers in this ISP. Note that the above performance bottlenecks are not specific to NR scheduling and may be even further aggravated in other scheduling schemes.

Note that the above performance bottlenecks are not specific to NR scheduling and are even more likely to occur with other schedulings (*e.g.*, random) as shown in Figure 76..

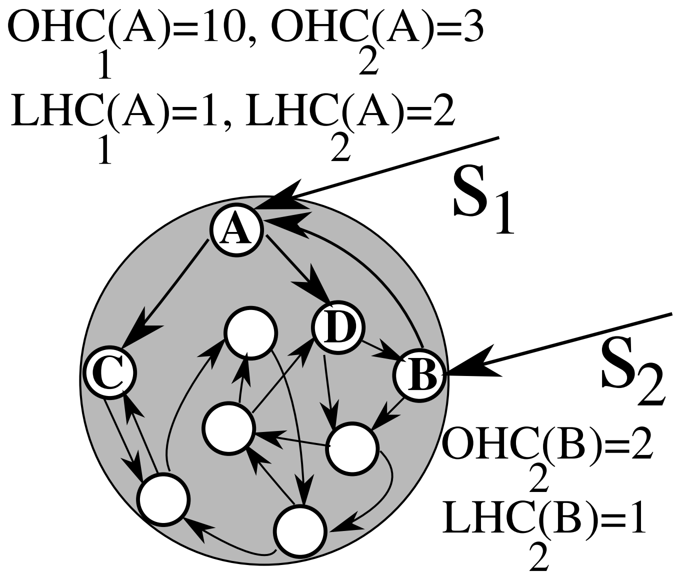


FIGURE 79.: An intra-ISP view of a localized overlay with associated hop counts for two substreams of S_1 and S_2.

7.6. OLIVES: Overview

In this section, we present OLIVES, a swarm-based P2P streaming protocol for live video. In OLIVES, participating peers in each ISP, maintain a fully localized overlay and incorporate an overlay-aware scheduling to ensure delivery of high quality stream. Since all connections have roughly the same bandwidth, the external traffic for individual ISPs is directly controlled by minimizing the number of incoming and outgoing external connections. Through this, ISPs can ensure that the aggregate incoming (and outgoing) external bandwidth is close to the stream bandwidth.

7.6.1. Maintaining a Localized Overlay

A *local tracker* within each ISP manages the internal and external connectivity of all peers within the ISP to ensure that overlay remains fully localized in the presence of churn. This also enables the ISP to enforce any policy for routing external traffic by directing external connections towards preferred ISPs (*e.g.*, [111, 113]) at the local tracker. Through a *global tracker*, local trackers discover external peers to establish external connections. When a peer with incoming external connection departs, the local tracker identifies a new external peer and prompts another local peer (with desired properties) to establish an external connection[3].

7.6.2. Overlay-Aware Scheduling

The main contribution of OLIVES is a two-tier overlay-aware scheduling scheme that maximizes the delivered quality to individual peers in a localized overlay. Motivated by the performance bottlenecks that we have identified in Subsection 7.5.1., content delivery in OLIVES is managed at two coherent or tiers as follows: *Inter-ISP Scheduling*: At this level, OLIVES focuses on the delivery of full-quality streams to individual ISPs. Inter-ISP scheduling is only concerned with external connections. *Intra-ISP Scheduling*: At this level, OLIVES ensures the delivery of each substream from edge peers to all internal peers. Intra-ISP scheduling is only responsible for managing

[3] We note that OLIVES can be used in a simpler setting where individual ISPs provide stable, provisioned servers that serve as edge peers.

internal connections. Since the main idea is to adopt NR scheduling at the two levels, in the OLIVES protocol each content block (or substream) carries the following three counters: *(i) ISP Hop Count (IHC)* keeps track of the number of ISPs that a block (or substream) has crossed. *(ii) Peer Hop Count (PHC)* keeps track of the number of internal peers that a block has visited within a single ISP. Therefore, this counter is reset by the corresponding edge peer where a block enters an ISP. *(iii) Overall Hop Count (OHC)* keeps track of the total number of peers (regardless of their ISP) that a block has visited. We use the following notations for the value of these counters for substream i at node p $IHC_i(p)$, $PHC_i(p)$ and $OHC_i(p)$.

7.7. Two-tier Overlay-Aware Scheduling

OLIVES periodically invokes inter- and intra-ISP scheduling in order to effectively utilize the bandwidth of individual connections despite the short- and long-term dynamics of congestion-controlled bandwidth. Each peer maintains an exponentially weighted moving average bandwidth ($bw(q)$) from each parent q and reports its newly available blocks along with the associated three hop counts and substream ID for each block to its children. Given the available blocks among its parents, each peer invokes the corresponding scheduling scheme to identify $n = \frac{bw(q)*\tau}{BlkSize}$ blocks to be requested from parent q where $BlkSize$ denotes the size of each block. Requested blocks from parents are sorted and thus delivered based on their timestamps.

7.7.1. Inter-ISP Scheduling

Since the connectivity among ISPs (*i.e.*, top-tier overlay) is random, NR scheduling can be adopted for inter-ISP scheduling to maximize the delivered quality to individual ISPs. To achieve this goal each ISP should behave as a single peer that implements NR scheduling by leveraging ISP hope count ($IHC()$). Considering each ISP as a single node, it learns about available blocks among its parent nodes and pulls the block with the smallest hop count (IHC) from each parent. In short, by leveraging IHC as hop count for each block, we can implement NR scheduling among ISPs in the top-tier overlay. The main challenge in inter-ISP scheduling, is to ensure that all of the incoming external connections of an ISP are coherently used to pull different blocks. Towards that, there should be a periodic coordination among incoming edge peers to ensure delivery of all blocks into an ISP without duplication. However, performing such a block-level coordination is prohibitively expensive. OLIVES incorporates two ideas to address this problem that will be discussed in the next two subsections.

7.7.1.1. Substream-level Coordination

We note that despite the variations in bandwidth of individual connections, the "ISP-level path" for individual blocks through the top-tier overlay is relatively stable because of the infrequent changes in the connectivity among ISPs. This implies that most of the blocks for a particular substream has a similar $IHC()$ value at a given external parent. OLIVES leverages this observation to perform substream-level coordination among incoming edge peers for mapping most of the blocks to external connections as follows: Each incoming edge peer uses the common value of IHC among available blocks of each substream s at its external parent q as its $IHC_s(q)$. The local tracker infrequently performs substream-level coordination among incoming edge peers in the following events: a peer is promoted to serve as an incoming edge peer, an existing incoming edge peer changes its external parent, or the common value of $IHC_s(q)$ for a substream changes. In such a coordination event, the tracker contacts each incoming edge peer to obtain the value of $IHC_s(q)$. The tracker runs NR scheduling based on $IHC_s(q)$ values, determines and reports the "designated substream" that should be pulled by each incoming edge peer. We emphasize that such a central coordination among the few incoming edge peers of each ISP is performed at the substream granularity and is triggered infrequently only when the connectivity among ISPs significantly changes. Therefore, the associated processing and communication overhead is small.

Figure 78. demonstrates the behavior of the proposed Inter-ISP scheduling in mapping both substreams to external connections by showing the values of their IHC at the outgoing edges of ISP_1. The inter-ISP scheduling in ISP_3 and ISP_4 maps substream S_1 to edge peers C and D to pull from their respective external parent. We examine the effect of inter-ISP scheduling on the delivered quality to individual ISPs through simulations using the same settings as discussed in Section 7.5.. Figure 80. presents the median delivered quality to individual ISPs as well as peers (along with

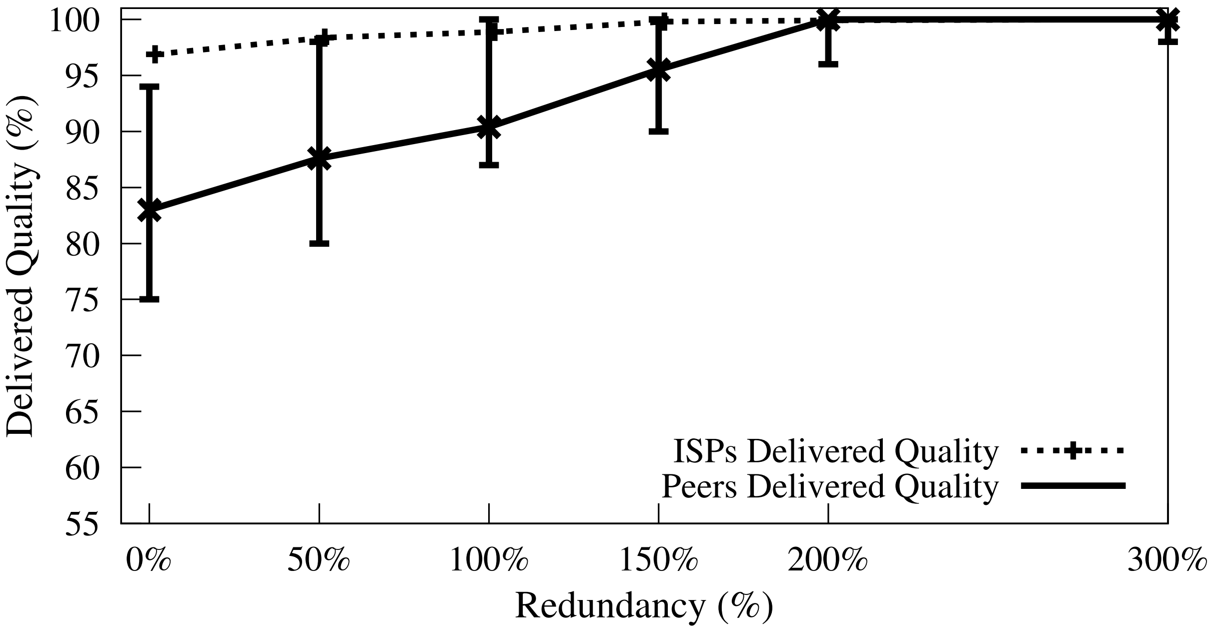


FIGURE 80.: Median of delivered quality with Intra-ISP scheduling to ISPs and peers.

the 10th and 90th percentile for peers) with the proposed inter-ISP scheduling as a function of the redundancy in external connectivity of individual ISPs. This result reveals that the inter-ISP scheduling significantly increases the delivered quality to individual ISPs over a fully localized overlay (*i.e.*, redundancy is 0). However, the minimum delivered quality to *individual peers* is still limited to 83% (Figure 80.) due to limitations in content delivery within individual ISPs, which we will address by intra-ISP scheduling.

7.7.1.2. Implicit Identification

Mapping substreams to external connections enables each incoming edge peer to identify a large fraction (or all) of the requested blocks of the designated substream from the external parent. After requesting all new blocks of the designated substream from the external parents, edge peers may have excess bandwidth on their external connection to request blocks from non-designated substreams. This requires some form of coordination among edge peers to divide responsibility for those blocks proportional to the edge peers' excess bandwidth. We note that since the aggregate bandwidth of incoming external connections for each ISP is equal or larger than the stream bandwidth, the aggregate excess bandwidth should be sufficient to pull all the unrequested blocks.

OLIVES leverages implicit identification to manage the blocks that have not been delivered to the ISP yet. Each incoming edge peer leverages the *unavailability of a block from non-designated substreams with sufficiently early timestamp among all of its internal parents (*i.e.*, its internal neighborhood) as an "implicit but reliable hint" that the block has not been requested by its corresponding edge peer.* To implement this idea, in each inter-ISP scheduling event, an edge peer with excess incoming external bandwidth examines the available blocks among its internal parents and identifies the largest timestamp for each non-designated substream s, as $ts_{max}(s)$. Then, it subtracts one interval τ from these maximum timestamps to identify a conservative maximum threshold ($ts_{th}(s)$ = $ts_{max}(s)$ - τ) for timestamp of blocks of each substream that must have been propagated to these internal parents by now. Blocks of a non-designated substream s with a timestamp lower than $ts_{th}(s)$ that are missing at all internal parents, are unlikely to have been requested by its designated edge peer for two reasons: *(i)* All requested (and thus delivered) blocks from parents are ordered based on their timestamps as we mentioned earlier, and *(ii)* Available blocks among internal parents represent the availability of content for a large fraction of peers due to the random connectivity within the ISP.

Once each incoming edge peer with excess bandwidth identifies unrequested blocks, it determines to request which unrequested block as follows: *Given the IDs of individual substreams, each incoming edge peer utilizes its excess bandwidth to pull the identified unrequested blocks in a prioritized fashion using a circular order (that is determined a prior) among the substreams.* For example, in a scenario with 4 substreams, the designated edge peer for pulling substream 3 uses its excess bandwidth to pull all identified missing blocks for substream 4, then for substream 1, and finally for substream 2. This method can effectively manage the utilization of excess bandwidth among edge peers (*i.e.*, it reduces the rate of duplicate blocks pulled into the ISP). Note that, each edge peer detects missing blocks of each non-designated substream s after certain delay which is proportional to the peer's distance from the corresponding edge peer of s. This adds a random delay to the reaction of edge peers and reduces the probability of duplication. Moreover, bandwidth deficit and surplus on connections are often short-lived and thus move among external connections.

7.7.2. Intra-ISP Scheduling

The goal of the intra-ISP scheduling is to deliver each block from the designated edge peer to all the internal peers in an ISP. In essence, each edge peer is treated as the designated source for the blocks that it pulls into the ISP. Therefore, OLIVES applies the idea of NR scheduling for individual blocks based on their relative local hop count from the corresponding edge peer (*i.e.*, LHC) (rather than total hop count from source). In each scheduling event (once per τ second), each internal peer considers the available blocks among its parents along with their LHCs and pulls the block with the minimum LHC from each parent.

Figure 79. demonstrates the behavior of the proposed intra-ISP scheduling by showing the average values of LHC for both substreams at peers A and B. In this case, peers C and D use LHC and pull blocks of substream S_1 from the edge peer A, regardless of the total hop count from the source which leads to the desired behavior. Simulating this scheduling reveals that peers delivered quality over a fully localized overlay reaches 95% as we will show in Section 7.8..

7.7.3. Buffer Requirement

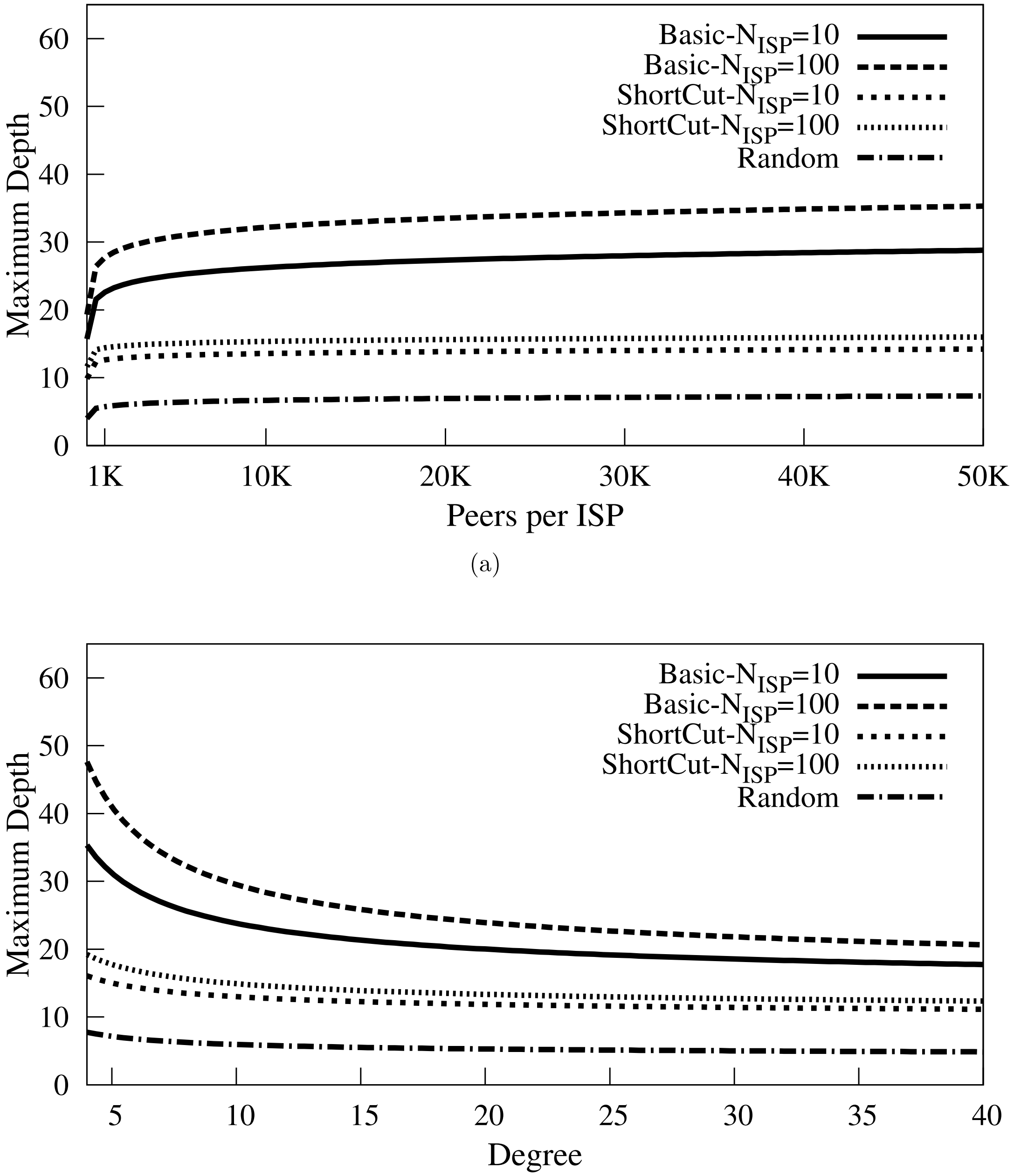
As the overlay connectivity becomes more localized, the delivery trees become inevitably deeper because there is limited flexibility in forming those trees over a localized overlay. The maximum depth of delivery trees (d_{max}) in OLIVES, is the product of two components: *(i)* the maximum depth of the *ISP-level* delivery trees, and *(ii)* the maximum depth of the delivery trees from each incoming edge peer to all internal peers within individual ISPs. Since the ISP-level overlay and connectivity within each ISP are both random, d_{max} in terms of hops can be derived by extending Eqn. VII.1 as follows:

$$(log_D^{(\frac{N\_ISP*(D-1)}{D}+1)} + 3) * (log_D^{(\frac{N*(D-1)}{D*N\_ISP}+1)} + 3) \qquad \text{(VII.7)}$$

Figure 81.(a) plots the maximum depth of delivery trees in a localized overlay and a comparable random overlay using Eqn. VII.7 as a function of the number of peers per ISP. As the figure shows the depth of delivery tree in OLIVES is always larger than the delivery tree over a comparable random overlay. OLIVES manages to deliver each substream to all ISPs despite limited inter-ISP connectivity in localized overlay at the cost of forming taller delivery trees. Taller delivery trees means more buffering at each peer.

7.7.3.1. Shortcutting of ISPs

The larger buffer requirement in OLIVES could be considered a drawback. One practical implication of this requirement is that addition of an ISP with a large number of peers to the overlay can significantly increase the buffer requirements for peers in other ISPs with small population. OLIVES adopts the idea of “shortcutting” to reduce this buffer requirement.



(a)

(b)

FIGURE 81.: Computed depth of delivery trees for localized overlay with and without shortcuts and a random overlay with the same properties. (a) and (b) depict the depth as a function of peers per ISP and Degree, respectively.

The basic idea is to minimize the distance between the incoming and outgoing external connections for each ISP by controlling the internal connectivity of each ISP. In OLIVES, each outgoing edge peer selects *all* incoming edge peers as parent as shown in Figure 82.. This mesh-like internal connectivity among all edge peers enables each outgoing edge peer to provide any substream to other ISPs even when mapping of incoming external connections changes by the coordination mechanism. Shortcutting has two opposite effects on the depth of delivery trees: First, it significantly reduces the depth ($OHC()$) of incoming edge peers of all ISPs on any delivery subtree. Second, it may slightly increase the distance between incoming edge peers and other internal peers of the corresponding ISPs on the delivery tree. This increase is at most one hop due to the logarithmic relation between depth within an ISP and its population. In summary, the overall effect of shortcutting leads to a significant decrease in the depth of delivery trees and thus buffer requirement. The maximum depth of the delivery trees with shortcuts can be derived as follows:

$$((log_D^{(\frac{N\_ISP*(D-1)}{D}+1)} + 2) * 2) + (log_D^{(\frac{N}{N\_ISP}-D)} + 3) + 1$$

Figure 81.(a) and 81.(b) shows the maximum depth of delivery trees in a localized overlay with shortcutting. This figure clearly demonstrates the ability of shortcutting to reduce depth of the delivery trees across the parameter space. Clearly, the cost of shortcutting is the overhead of maintaining the connectivity among edge peers in the presence of churn which can be performed by the local tracker.

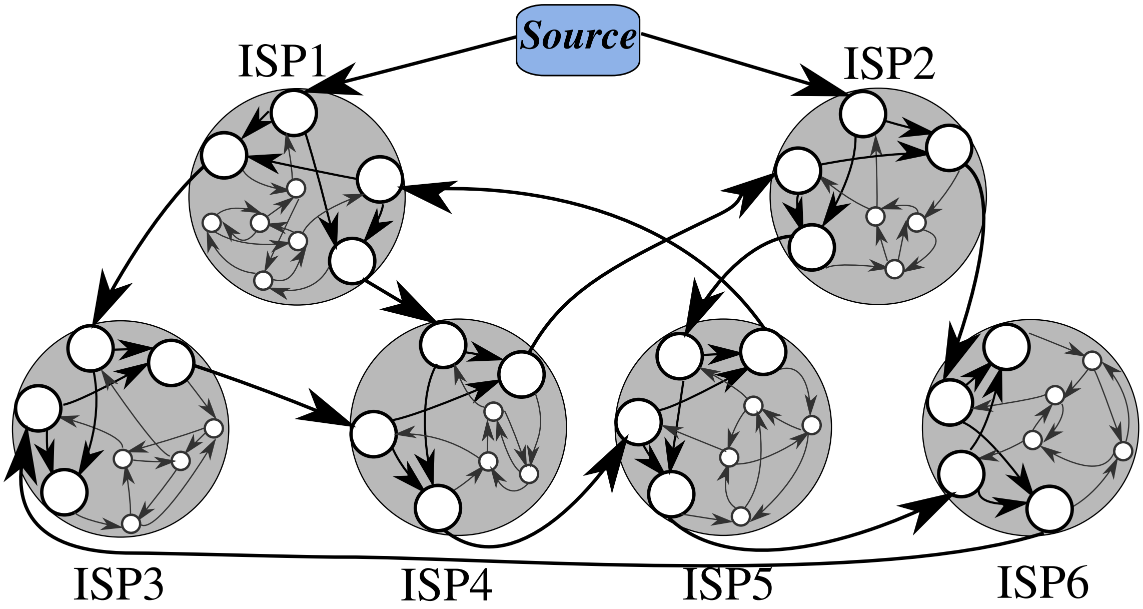


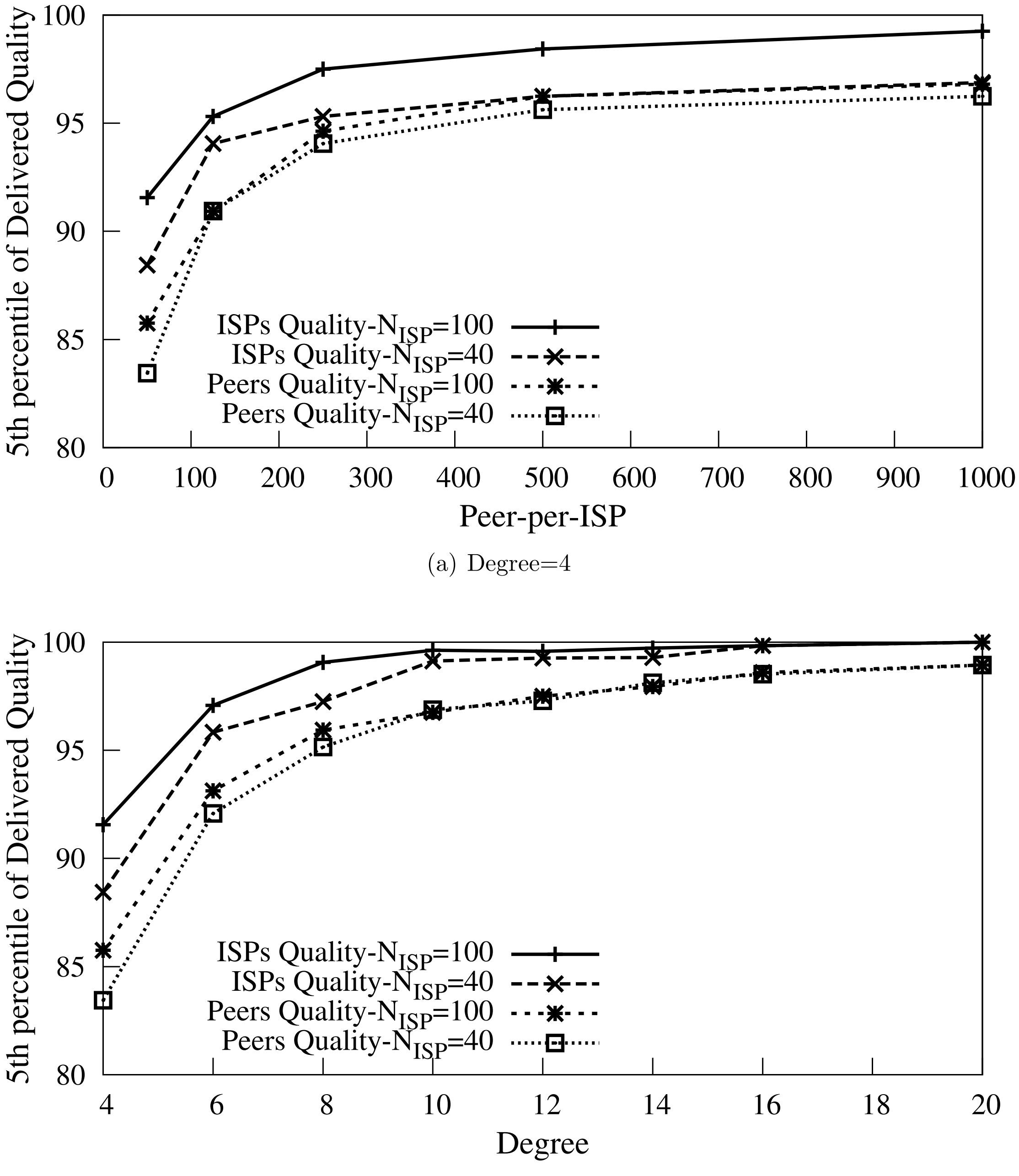
FIGURE 82.: A localized overlay with shortcuts.

7.8. Performance Evaluation: Overlay Connectivity

In this section, we examine how various overlay properties affect the overall performance of two-tier scheduling in OLIVES using substream abstraction for content delivery. Towards this end, we only focus on fully localized overlay where incoming and outgoing peer and ISP degree are equal to the number of substreams. The effect of bandwidth dynamics are examined in the next section.

7.8.1. Peer degree, ISP & Peer Population

We start by examining the effect of the following three parameters that primarily determine the overall connectivity of an overlay: peer degree, number of ISPs and the number of peers. Figure 83.(a) depicts the 5th percentile of delivered quality to



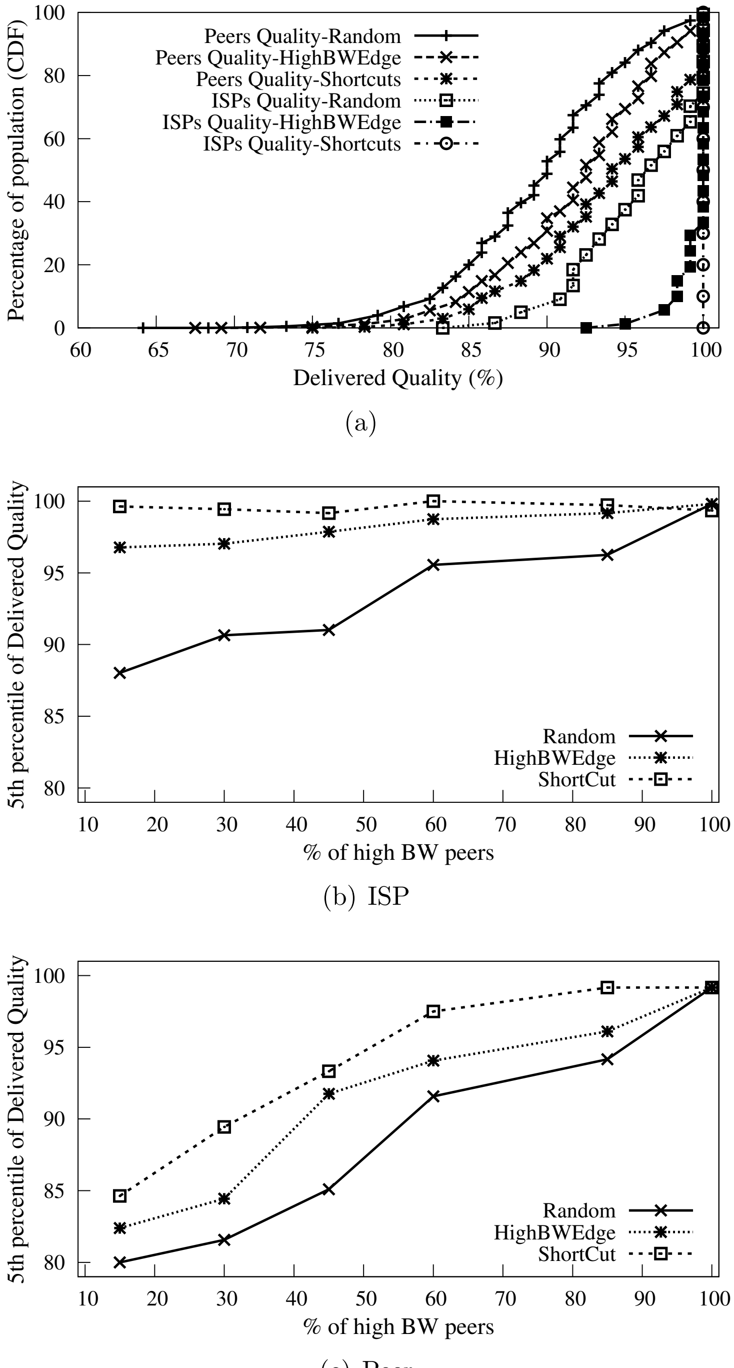
(a) Degree=4

(b) Peer per ISP=50

FIGURE 83.: Effect of peer degree, ISP and peer population: (a) and (b) depict the 5th percentile of delivered quality to Peers and ISPs, with degree of 4 and peers per isp of 50, respectively.

individual peers and ISPs as a function of peer population per ISP. The incoming and outgoing degree of all peers and ISPs is 4. Each line shows the results for a certain number of ISPs, namely 40 and 100 ISPs. Showing the 5th percentile of delivered quality indicates that 95% of peers or ISPs in each scenario received a higher quality than the shown value. This figure shows that increasing the number of peer per ISP as well as ISPs in the overlay initially improves the performance. To explain this, we note that as the population of nodes in a graph increases, the graph becomes less "clustered". This effect is more pronounced when node degree is small. The lower level of clustering provides more flexibility for delivery trees to reach all nodes and thus results in higher delivered quality. Figure 83.(a) demonstrates this phenomenon both in the ISP-level overlay and within each ISP.

Figure 83.(b) depicts the 5th percentile of delivered quality to peers and ISPs as a function of peer degree when the number of peers per ISP is 50. Each line shows the result for a different number of ISPs in the overlay. This figure clearly shows that increasing peer degree improves delivered quality. Increasing peer degree improves the overlay connectivity at both levels which facilitates the formation of intra- and inter-ISP delivery trees by the scheduling. *In summary, the two tier scheduling in OLIVES exhibits a good performance in most combinations of peer degree, peer per ISP and ISP per overlay. The performance is moderately dropped only in corner scenarios where all three parameters are small.*



(a)

(b) ISP

(c) Peer

FIGURE 84.: Effect of bandwidth heterogeneity: (a) Distribution of delivered quality to ISPs and high bandwidth peers. There are 85% low and 15% high bandwidth peers. (b) and (c) the 5th percentile of delivered quality to peers and ISPs, as a function of the percentage of high bandwidth peers, respectively.

7.8.2. Heterogeneous Peer Bandwidth

We now turn our attention to overlays with heterogeneous (but symmetric) peer bandwidth. In particular, we consider an overlay with 20,000 peers that are evenly grouped into 40 ISPs. Degree of high and low bandwidth peers are 12 and 6, respectively. We vary the percentage of high bandwidth peers from 15% to 85%. We assume that the stream is MDC encoded and the delivered quality to each peer is proportional to its incoming bandwidth. We focus on delivered quality to high bandwidth peers as they should receive all substreams whereas low bandwidth ones should only receive half of the substreams. Figure 84.(a) shows the distribution of delivered quality to individual peers and ISPs for three different overlay construction strategies:

(i) Random: random peers are selected as edge peers,

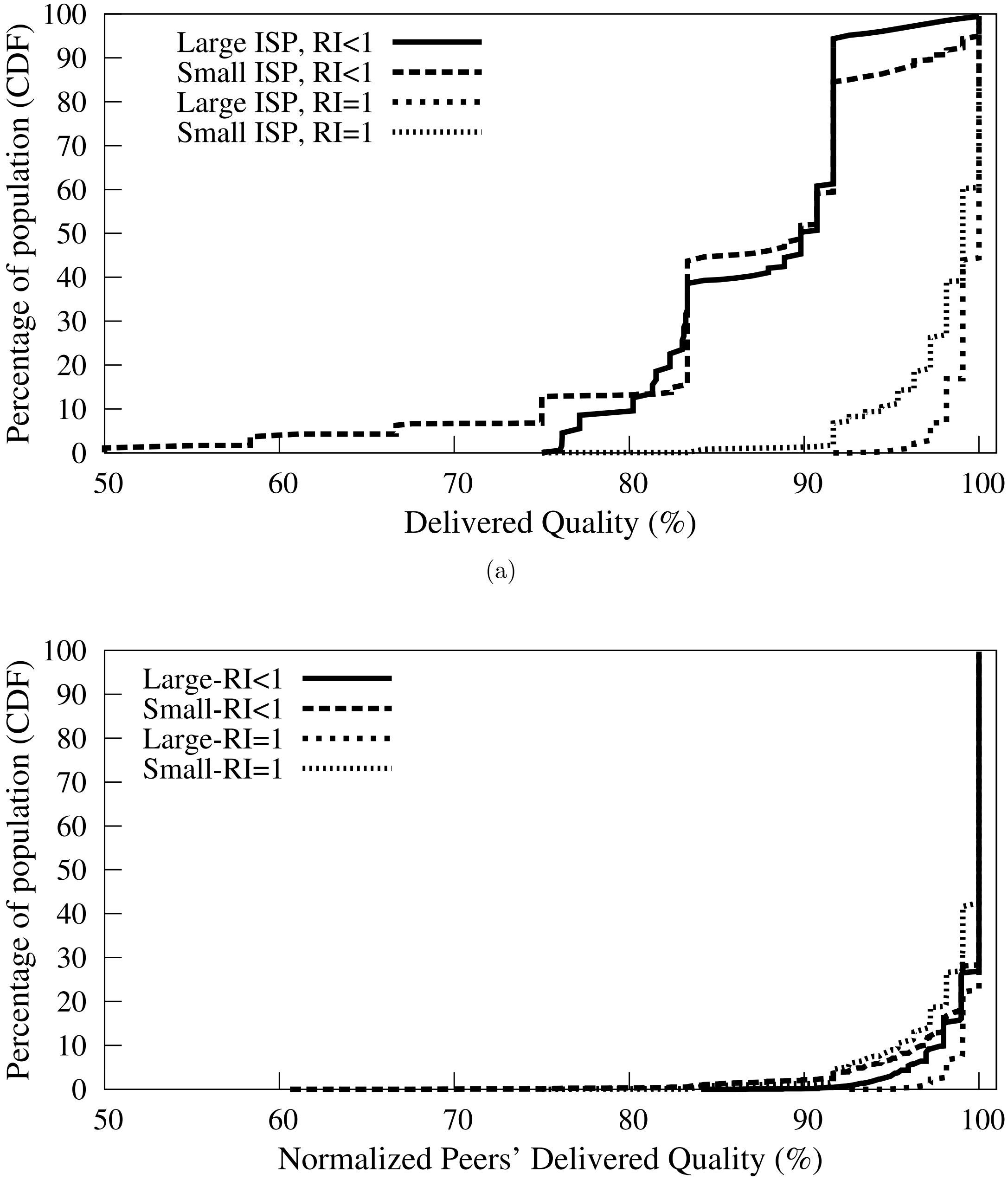
(ii) HighBWEdge: high bandwidth peers are selected as edge peers, and

(iii) Shortcuts: high bandwidth peers are selected as edge and shortcutting is used.

In Figure 84.(a) the percentage of high bandwidth and low bandwidth peers are 15% and 85%, respectively. Figures 84.(b) and 84.(c) show the 5th percentile of delivered quality to individual ISPs and peers as a function of the percentage of high bandwidth peers, respectively. Figure 84.(b) indicates that randomly placing peers as edges reduces the delivered quality to some ISPs. The key problem is *when a high bandwidth peer has one or more low bandwidth parents, it becomes more difficult for the scheduling to map the required substreams among the parents because* *every substream parents.* With “Random” strategy, the problem with low-bandwidth parents occurs in both inter- and intra-ISP schedulings which reduces the delivered quality to ISPs. “HighBWEdge” strategy eliminates the problem between edge peers and significantly improves delivered quality to ISPs. However, since the relay of each substream through individual ISPs is determined by the intra-ISP scheduling, the problem with low bandwidth parents within each ISP still affects the intra-ISP scheduling as can be seen by the 5th percentile delivered quality to ISPs in Figure 84.(b) for “HighBWEdge”. Shortcutting eliminates this latter problem and maximizes the delivered quality to all ISPs. The delivered quality to peers is lower than ISPs since high bandwidth peers may still have internal low bandwidth parents. Moreover, increasing the percentage of high bandwidth peers, reduces the probability of having a low bandwidth peer as an edge peer or internal parent which leads to a higher delivered quality as shown in Figures 84.(b) and 84.(c).

7.8.3. Resources & Skewed ISP Population

To examine the effect of the amount of resources in the system, we consider a scenario from real P2P application traces. Towards this end, we use a sample snapshot of Gnutella application crawled in July 2009. The snapshot consists of 50K peers (represented by an IP address). Number of ASes and the population of peers per ASes are derived by mapping the IP address of the crawled peers to ASes. The snapshot consists of 970 ASes (or ISPs) with skewed distribution of peers per ASes



(a)

(b)

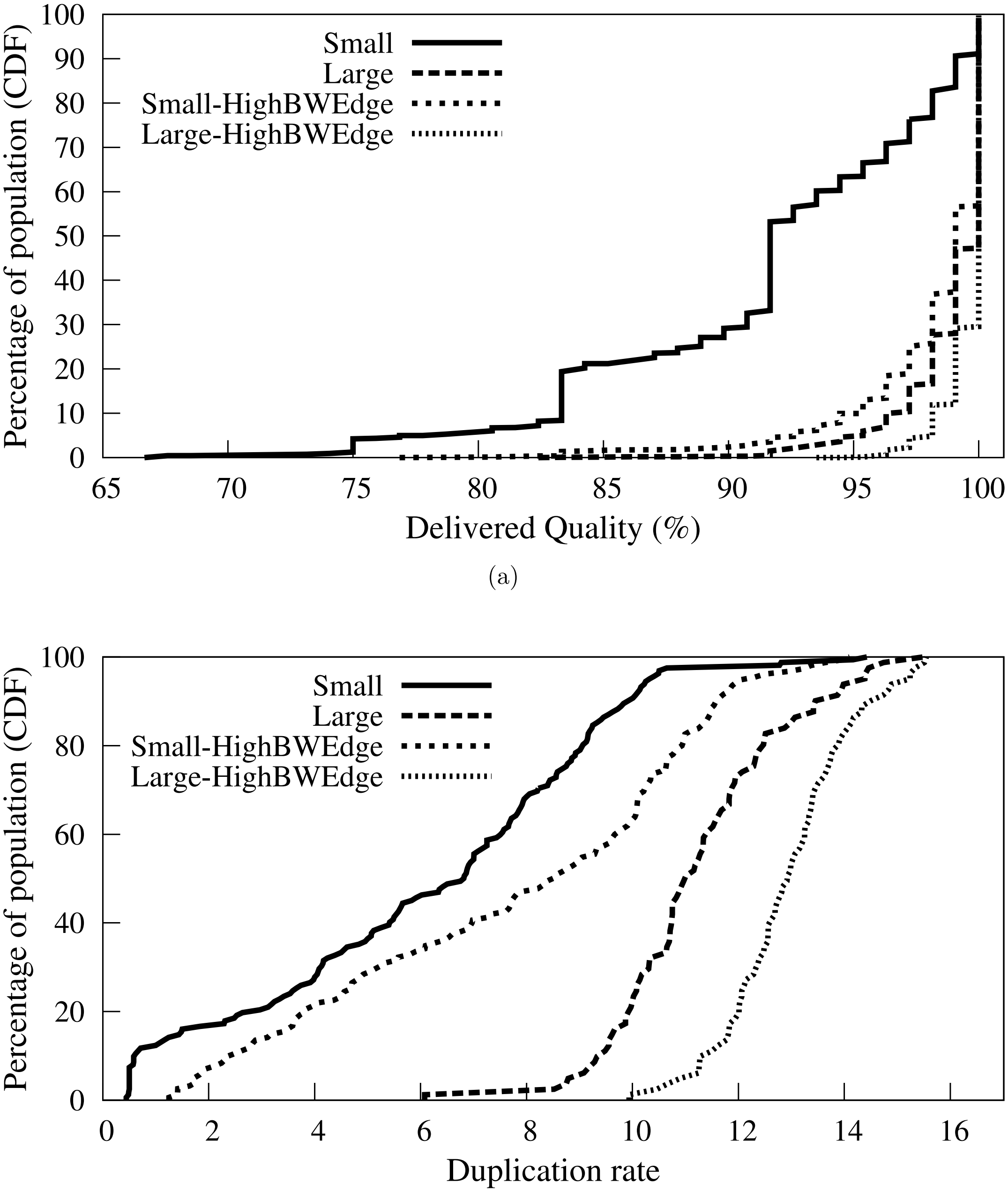
FIGURE 85.: Effect of resources and skewed ISP population: (a) and (b) Distribution of delivered quality and normalized delivered quality to peers, respectively.

(85% of peers are in 10% of ASes). We assume the incoming bandwidth of peers is enough to receive the full quality stream and the stream bandwidth is 530 Kbps. The outgoing bandwidth of 15%, 35% and 50% of peers is set to 128Kbps, 384Kbps and 768Kbps, respectively derived from [137]. We set the in-degree of peers as 12, and adjust the peers outgoing degree according to their outgoing bandwidth, *i.e.*, outgoing degree of peers with 128Kbps, 384Kbps, 768Kbps is set to 3, 9 and 18, respectively.

To examine the effect of resource index (RI), we let 20% of ISPs be resource deficit while their resource index is randomly chosen between 0.7-0.95. Figure 85.(a) depicts the distribution of delivered quality to peers in small (population < 100) and large ISPs (population ≥ 100) with RI < 1 or RI $= 1$. This figure reveals that regardless of the size of the ISPs, peers in the ISPs with bandwidth deficit experience a lower quality. Note that, ISPs' delivered quality is almost 100% across all ISPs (not shown). In order to verify the determining factor for low delivered quality, we plot the percentage ratio of delivered quality to peers normalized by their incoming access link bandwidth utilization (representing their number of parents) in Figure 85.(b). Based on Figure 85.(b) we can conclude that the major bottleneck for delivered quality is the amount of available incoming bandwidth to each peer as normalizing the quality to the number of parents for each peer shows a roughly similar performance across all peers in all ISPs.

7.8.4. Asymmetric Peer Bandwidth

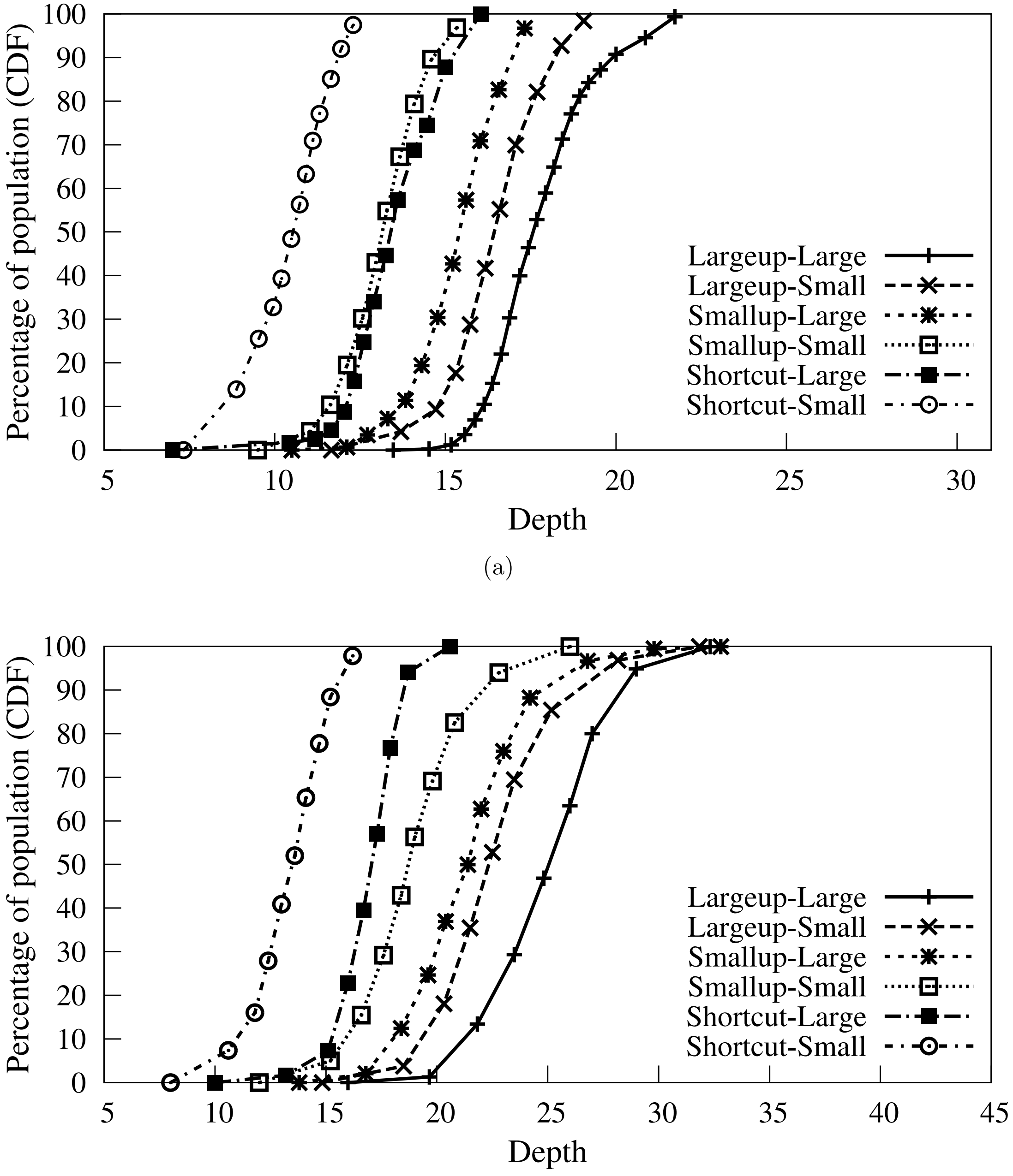
To investigate the effect of bandwidth asymmetry, we reuse the real P2P trace and bandwidth distribution that described in Subsection 7.8.3. and ensure that the resource index of each ISP is 1. Figure 86.(a) depicts the distribution of delivered quality to individual peers in small (population le 100) and large ISPs (population $>$ 100). Intra and Inter-ISP connectivities are random or peers with high outgoing bandwidth are selected as an incoming edge (lines labeled by “*-HighBWEdge”). Figure 86.(a) shows that the delivered quality to peers in small ISPs for the random overlay is significantly lower than large ISPs. However, promoting only peers with high outgoing bandwidth as an edge, increases the delivered quality to the peers in small ISPs. Note that, as the number of incoming edge peers of each ISP is similar, the probability of an incoming edge peer being connected to another incoming edge peer is inversely proportional to the population of the ISP. Due to intra-ISP scheduling, children of edge peers are pulling the substream that their corresponding parent receives from its external connection, thus, for an edge peer a with designated substream s and d children, the possible number of peers that can provide substream s inside the ISP will be the equal to d only if none of the d children are edge peers, otherwise, it is equal to non-edge children of a. In small ISPs, as the probability of having an edge child increases for each edge peer, some substreams are duplicated in the ISP at a lower rate which results in more difficulty in construction the delivery tree for those substreams inside the ISP. Figure 86.(b) depicts the average duplication



(a)

(b)

FIGURE 86.: Effect of asymmetric peer bandwidth: (a) and (b) Distribution of delivered quality to peers while random or high bandwidth peers are chosen as edges for small and large ISPs. (b) Distribution of duplication rate for the same scenarios as in (a).



(a)

(b)

FIGURE 87.: Effect of location of small and large ISPs: (a) and (b) Distribution of average and maximum depth of peers in overlays with various criteria for placing ISPs with different sizes, respectively.

rate for small and large ISPs in random and HIGHBWEdge overlays. The figure reveals that for small ISPs the duplications rate is much lower than large ISPs and its median increases by 2 when only high bandwidth peers are selected as edge peers which increases the delivered quality to peers.

7.8.5. Location of Small and Large ISPs

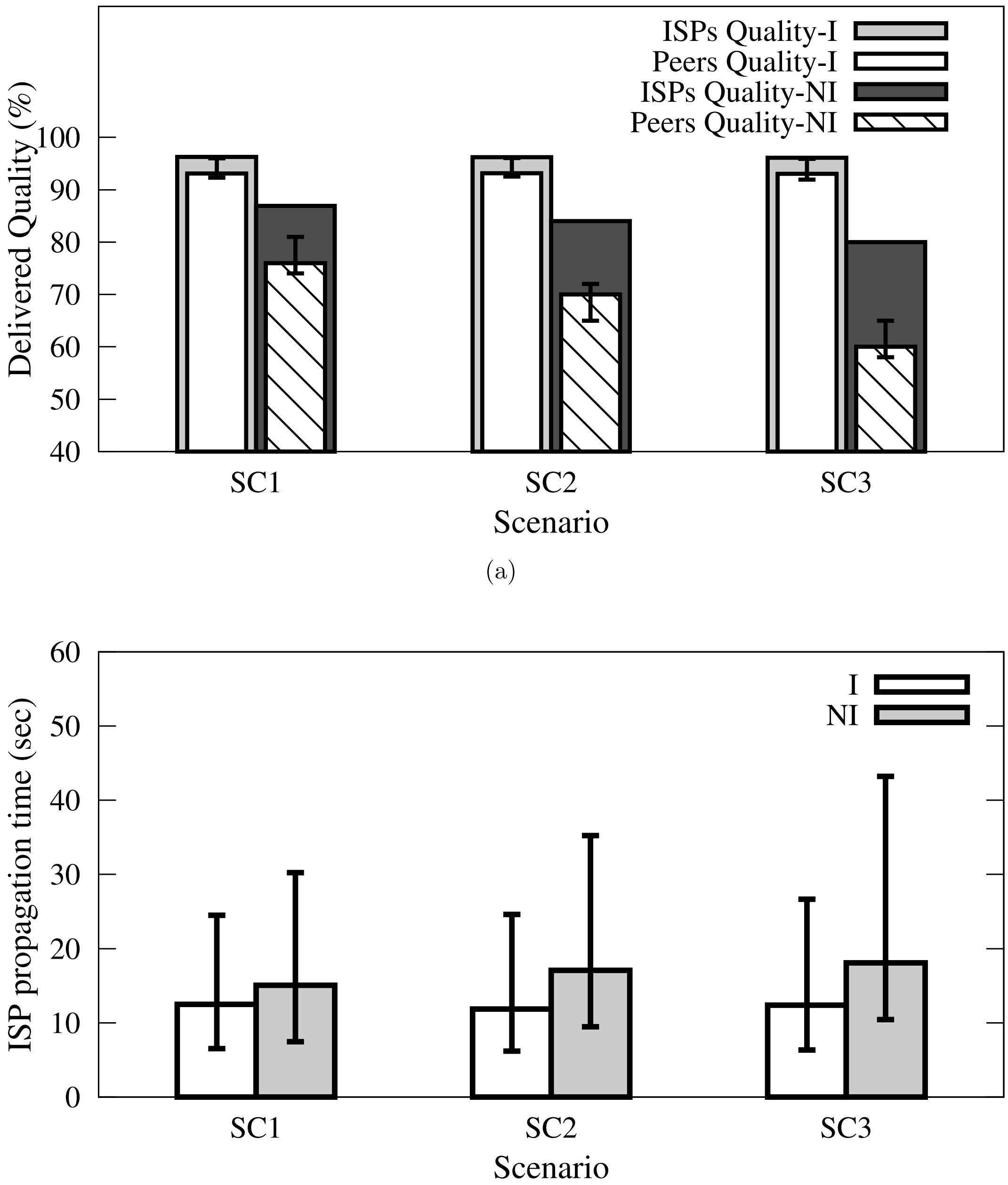
Our goal is to investigate the effect of the location of large ISPs on the overall depth of delivery trees across peers in small and large ISPs. Towards that, we use the real P2P trace and bandwidth distribution that described in Subsection 7.8.3., while ensuring that the resource index of each ISP is 1.

We examine two scenarios of placing large ISPs and small ISPs close to source which we call "Largeup" and "Smallup", respectively. Figure 87.(a) and 87.(b) depict the distribution of average and maximum depth of peers across delivery trees in small and large ISPs for the above scenarios and shortcut overlay, respectively. These figures illustrate that while the average and maximum depth of delivery trees are generally lower for peers in smaller ISPs, in Largeup scenario, the average and maximum depth of both groups of peers, is larger than Smallup and shortcut scenarios. In essence, if large ISPs are positioned close to source, all the delivery trees of all peers cross through large ISPs. The relative distance between the incoming and outgoing edge peers in large ISPs can be much larger than small ISPs. Therefore, in Largeup scenario in which large ISPs are the ancestor of all peer in all delivery trees, the depth of delivery trees increases. On the other hand, in Smallup scenario, large ISPs are in a random distance (>1) from source and are the ancestor of some but not all of the other ISPs in the Inter-ISP overlay delivery trees. Thus, in Smallup scenario, in average the depth of delivery trees reduces compared to Largeup scenario. Figures 87.(a) and 87.(b) also show the depth of delivery trees for shortcut overlay. From these figures we can clearly observe two interesting points: First, the maximum depth of delivery trees (and thus buffer requirement at each peer) for both group of peers has decreased in shortcut overlay compared to Smallup and Largeup scenarios. Second, there is a clear gap between maximum depth of delivery trees in both group of peers (shown in Figure 87.(b)). Peers in smaller ISPs typically have a much smaller maximum depth than peers in large ISPs, and thus require proportionally less buffering. *Through shortcutting, peers in smaller ISPs typically have a much smaller maximum depth than peers in large ISPs, and thus require proportionally less buffering.*

7.9. Performance Evaluation: Bandwidth & Peer Dynamics

In this section, our main goal is to evaluate the performance of OLIVES intra- and inter-ISP scheduling in presence of bandwidth and peer dynamics. Towards this end, we use *ns2* to conduct packet level simulations. This allows us to construct various scenarios and reliably identify underlying performance bottlenecks. We consider an overlay with 101 ISPs where one ISP, called target ISP, has 100 heterogeneous peers but all other ISPs are simulated as a single peer. This enables us to examine the effect of packet level dynamics on the performance of content delivery to a single ISP within the size of feasible scenarios in packet level simulators. 85% of peers in the target ISP, have low (750Kbps) and the rest have high (1.5Mbps)symmetric access link bandwidth. Peers in these two groups maintain (incoming and outgoing) degree of 10 and 20, respectively. The video stream has a bandwidth of 1.5 Mbps and is MDC-encoded with 10 descriptions of 150 Kbps. Therefore, low and high bandwidth peers should receive 5 and 10 descriptions, respectively. Source bandwidth is set to 1.6 Mbps to ensure the delivery of full quality stream (with minimal redundancy) to its 20 children despite any packet loss. All connections are TCP-friendly congestion controlled. The physical topology is generated by Brite [126] with 15 ASes and 10 routers per AS in top-down mode[4]. We focus on the delivered quality to high bandwidth peers since low bandwidth peers receive full quality stream (proportional to their incoming bandwidth) in all scenarios. The interval for periodic scheduling (τ) is set to 6 seconds, however, the presented results are not sensitive to the choice of the scheduling interval. Each simulation is run for 2000 simulated seconds and the presented results are averaged over 10 runs with different random seeds.

[4] Peers in the target ISP are randomly mapped on the physical topology in order to have a combination of high and low bandwidth connections within the target ISP and among ISPs.



(a)

(b)

FIGURE 88.: Effect of per-connection bandwidth heterogeneity: (a) Delivered quality to ISPs and peers in 3 scenarios of RTT heterogeneity with (I) and without (NI) implicit identification. (b) Propagation time to the ISP in the 3 RTT scenarios with (I) and without (NI) implicit identification.

7.9.1. Per-Connection Bandwidth Heterogeneity

We increase the diversity of average congestion controlled bandwidth across different connections by controlling the range of RTT values. Towards this end, we consider three reference scenarios $SC1$, $SC2$ and $SC3$ by randomly selecting the delay on each access link from the following ranges [5ms, 25ms], [5ms, 100ms], and [5ms, 150ms], respectively. Figure 88.(a) shows the delivered quality to the target ISP, and the median (and bars for 5th and 95th percentiles) delivered quality to its high bandwidth peers with implicit identification mechanism (labeled as "I") and without it (labeled as "NI") [5] in all three reference scenarios. This figure reveals that the implicit identification mechanism can deliver high quality stream to all peers despite the increasing level of heterogeneity of average bandwidth among overlay connections. However, in the absence of implicit identification, the delivered quality to the ISP shows a moderate decrease while its gap with the delivered quality to high bandwidth peers quickly widens with the level of bandwidth heterogeneity. Furthermore, the percentage of duplicate blocks that are pulled into the ISP can be effectively limited below 1.2% with implicit identification but it varies between 9% to 13% without it.

To explain this, Figure 88.(b) depicts the median (and bars for 5th and 95th percentiles) time between the generation of a block at source and its first arrival at an incoming edge peer in the target ISP (*i.e.*, propagation time) for the three scenarios.

[5] For a reasonable comparison, in the absence of coordination, the inter-ISP scheduling at edge peers with excess bandwidth identifies missing blocks among its internal parents and pulls a random subset of these missing blocks.

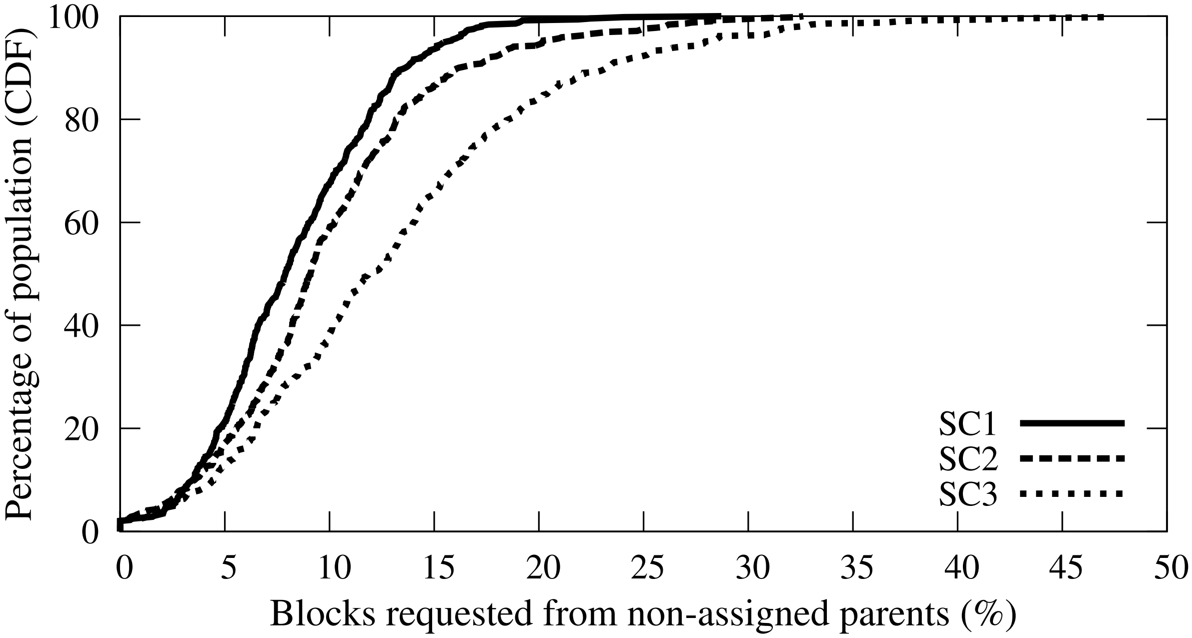


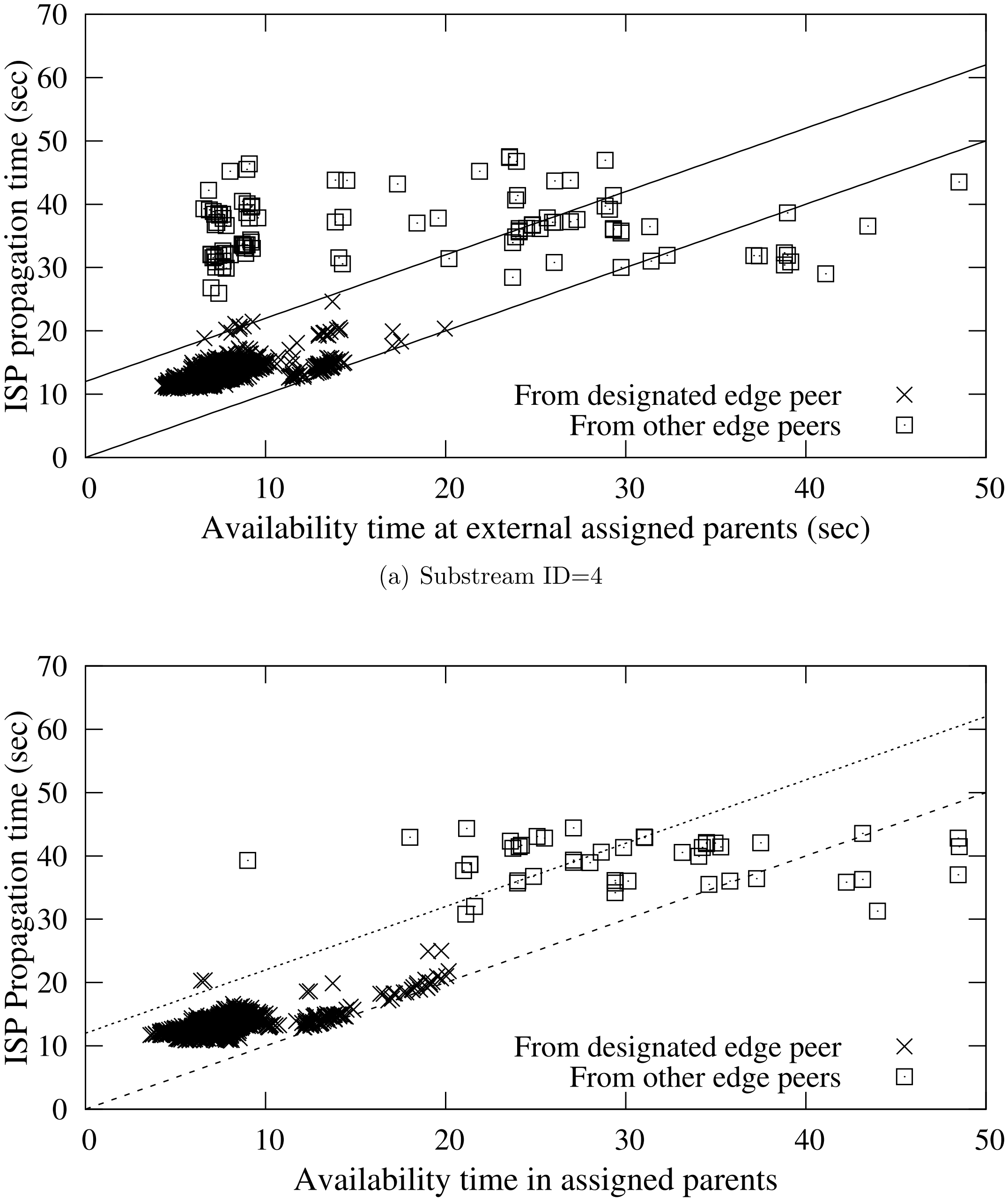
FIGURE 89.: Distribution of blocks requested from non-assigned parents for 3 scenarios.

Figure 88.(b) indicates that in the absence of identification, blocks experience a longer propagation time, and this difference further grows with the heterogeneity of per connection bandwidth. *Overall, these results shows that despite large variations in connection bandwidth, the implicit identification in inter-ISP scheduling can effectively utilize excess external connection bandwidth by pulling the missing blocks in a timely manner.*

7.9.2. Behavior of Implicit Hint & Coordination Mechanism

Figure 89. depicts the distribution of blocks requested from non-assigned parents for the above three scenarios. On average 7% of blocks across all substreams are pulled into the ISP by a non-designated edge peer in scenario *SC1*. This number increases to 9% and 14% in scenarios *SC2* and *SC3*, respectively. We take a closer look at the micro-level dynamics of the implicit identification. Figures 90.(a) and 90.(b) are scatter plots of the propagation time of each block to an external parent of target ISP (as x axis) vs its propagation time to an incoming edge peer that first pulls the block (as y axis) for all blocks of two sampled substreams of 4 and 8 in scenario SC3, respectively. Blocks that enter the ISP through their designated edge are marked with an "X". These two figures illustrate the relative time for availability of a block outside and inside the target ISP. Roughly 90% of blocks in these two substreams are pulled into the ISP by the designated edge peer as soon as they become available at the corresponding external peers. Figures 90.(a) and 90.(b) clearly show that the gap between the propagation time of these blocks outside and inside the ISP is very small (all points are between lines of $y = x$ and $y = x + 2 * \tau$).

Blocks that are pulled into the ISP by non-designated peers can be divided into two groups: First, those blocks that quickly become available at the designated parent but they are requested through other external parents after some delay. These blocks were not requested from the designated parent due to the short-term bandwidth



(a) Substream ID=4

(b) Substream ID=8

FIGURE 90.: Behavior of implicit hint and coordination mechanism: (a) and (b) Scatter plots of blocks propagation time to the ISP for two different substreams with ID 4 and 8, respectively.

deficit of the corresponding external connection. Second, those blocks that became available rather late at the designated parent.

Figure 90.(a) reveals an example of a connection with lower bandwidth than the substream bandwidth as most of the blocks that are requested from non-assigned external parents are also available within a short time ($<$ 10 sec) in the assigned external parent. The only reason that these blocks are not requested from the assigned parent is due to the low bandwidth connection between edge peer and the corresponding external parent. On the other hand, Figure 90.(b) is an example of a connection that suffers content bottleneck as most of the blocks that are requested from non-assigned external parents are available at the assigned external parent much later than the rest of the blocks which results in holes in the sequential block requests of this particular substream and invokes the optimized coordination mechanism by other edge peers.

Figure 91. depicts the distribution of time that blocks of a particular substream (*i.e.*, substream 9) becomes available in the assigned external parent, ISP propagation time of the subset of those blocks that are requested from the assigned external parent and the ISP propagation time of the rest of the blocks requested from non-assigned external parents. As the figure depicts, within one δ (*i.e.*, 4 sec) of the availability time in the assigned external parent, blocks will be delivered in the ISP through the assigned external parent. The portion of blocks that has been requested from non-assigned external parents are delivered much later in the ISP (*i.e.*, more than

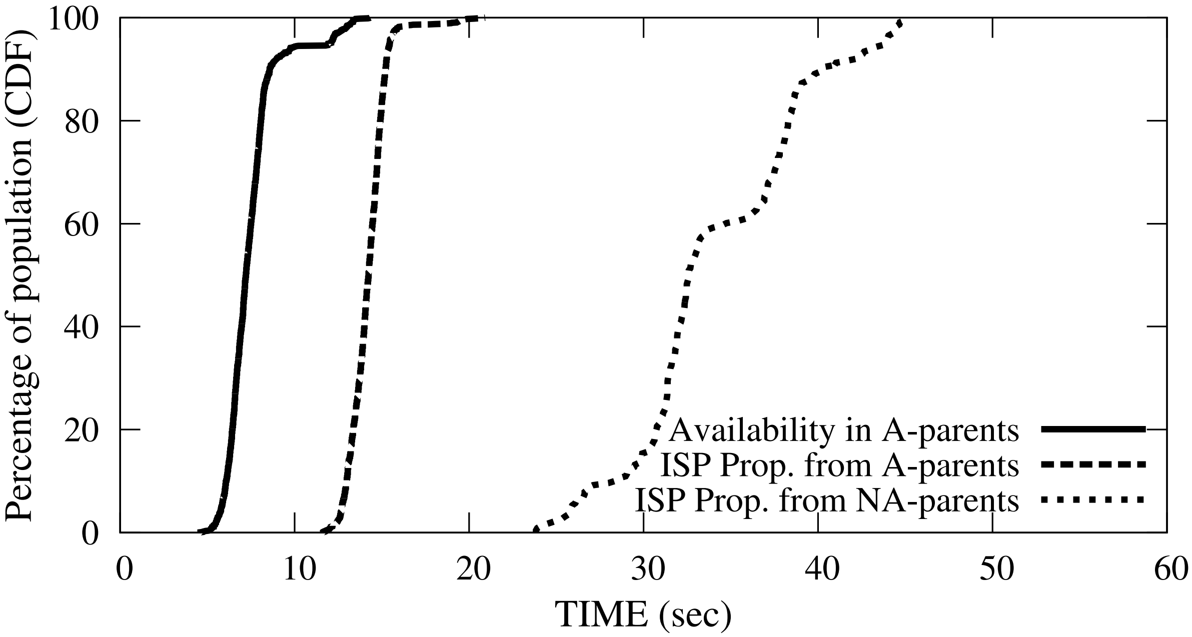


FIGURE 91.: Behavior of implicit hint and coordination mechanism: Distribution of block availability time in assigned parent and ISP propagation time from assigned (labeled as 'A') and non-assigned (labeled as 'NA') parents in a sample substream with ID 9.

$2*\delta$ sec). As we have described before, the determining factor for the ISP propagation time of blocks requested from non-assigned external parents is the overlay distance of the edge peers. Therefore, in the subtree 9 which is shown in Figure 91., the minimum distance of all edge peers from the responsible edge peer of subtree 9, is 2.

7.9.3. Peer Dynamics

To evaluate the performance of OLIVES in the presence of peer dynamics, we incorporate churn in the three reference scenarios using the churn model reported in

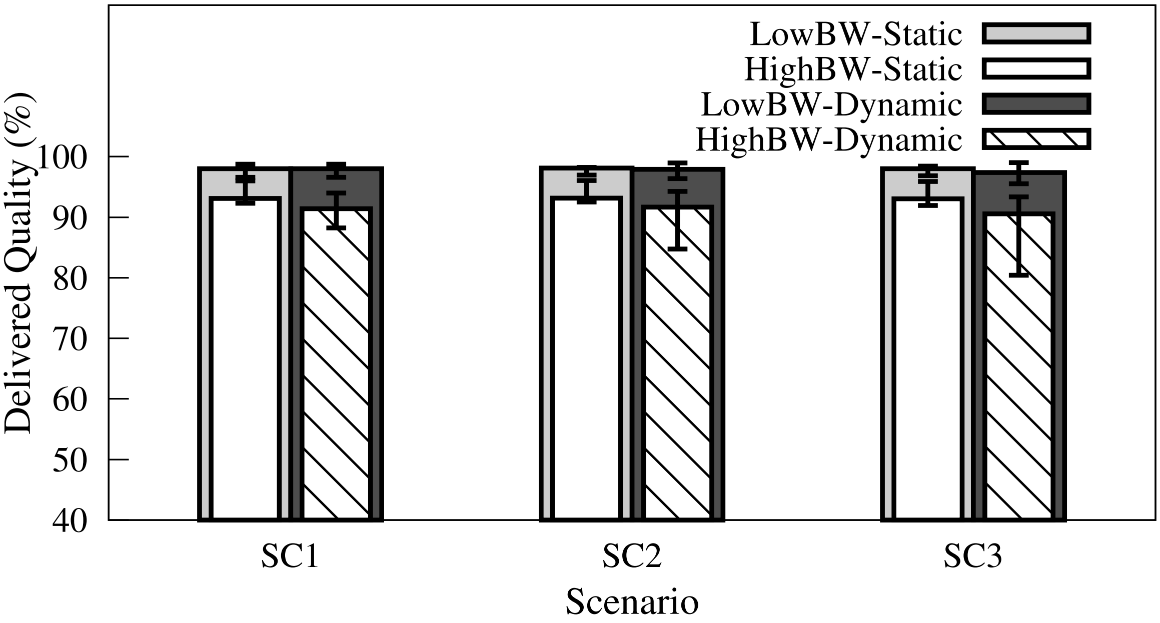


FIGURE 92.: Effect of peer dynamics: 5th, 95th and median of delivered quality to high and low bandwidth peers.

empirical studies on deployed P2P streaming systems [129, 87]. Towards this end, we select peer session times from a log-normal distribution (μ=4.29 and σ=1.28), and peer inter-arrival times from a Pareto distribution (a=2.52 and b=1.55). In the presence of churn, the aggregate average incoming bandwidth to the target ISP is affected by the performance of the peer discovery mechanism to identify external peers. To examine OLIVES without relying on a particular discovery mechanism, we increase the external degree of the target ISP by 10% so that the aggregate incoming bandwidth in presence of churn is roughly the same as stream bandwidth.

Figure 92. depicts the median (and bars for 5th and 95th percentiles) delivered quality to low and high bandwidth peers in the target ISP with and without churn labeled as “dynamic” and “static”, respectively. This figure indicates that the performance of low bandwidth peers is not affected by churn. The median and 95th percentile of delivered quality to high bandwidth peers in presence of churn are very similar to the static setting across all scenarios. However, the delivered quality to a small fraction of high bandwidth peers (5th percentile) slightly decreases in presence of churn and further decreases in scenarios with larger per connection bandwidth heterogeneity (*i.e.*, SC3). Closer examination reveals that the main contributing factor for slightly lower delivered quality with larger connection bandwidth heterogeneity is the implicit identifications in determining the non-requested blocks.The change in the delivered quality depends on *(i)* the aggregate time for replacing any departing peer, and *(ii)* the behavior of coordination mechanism with high rate of change in parents. The time it takes to replace a departed edge peer or find another external edge peer could affect the aggregate delivered quality to each ISP. However, a major determining factor for lower delivered quality in higher RTT heterogeneity scenarios is due to the implicit identifications in determining the unrequested blocks. In presence of churn, some parent peers might have recently joined the session, and thus, the block availability among all parents can give an incomplete view of the total internal block availability. Essentially, this can result in an inaccurate identification of the non-requested blocks which leads to requesting more duplicate blocks (2%, 5% and 6% for SC1, SC2 and SC3, respectively). *Overall, the resulting change in the delivered quality due to churn is minimal (less than 10%) which shows the ability of the implicit identification mechanism to achieve good performance even in presence of peer dynamics.*

7.10. Summary

In this chapter, we investigated the design and evaluation of an ISP-friendly P2P streaming mechanism for live content. We examined the performance of commonly used P2P streaming applications over localized overlays and identified fundamental underlying reasons that adversely affect the performance of such applications. Based the above insights, we designed a new Overlay-aware LIVE P2P Streaming mechanism called OLIVES that incorporates a two-tier block scheduling scheme over maximum localized overlays to overcome the constraints imposed by localization. Through detailed simulations we evaluated the performance of OLIVES and demonstrated its ability to achieve good performance over a wide range of realistic scenarios while maximizing the traffic localization. We believe our work provides valuable insights in the behavior of P2P streaming applications over localized overlays.

As we have discussed in this chapter, Section 7.3., there are three various approaches to deal with P2P traffic localization in the context of live streaming. In this chapter, we presented our work on P2P localization which adapted a hybrid approach by revising both the overlay (making it localized) and block scheduling.

In the next chapter, we present our ongoing work on P2P traffic localization which adapted a different approach, namely, revising the overlay connectivity.

CHAPTER VIII

OVERLAY MONITORING & REPAIR IN MESH-BASED P2P STREAMING

Material in this chapter was adopted from a paper published [7] in the *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)* at June 2009. This work is co-authored with Prof. Reza Rejaie who provided technical guidance. The experimental work is entirely mine. The text is written jointly by myself and Prof. Reza Rejaie.

The connectivity of the overlay plays a key role on the performance of content delivery in mesh-based P2P streaming mechanisms. In particular, random nature of connections ensures the diversity of content among connected peers. This in turn enables individual peers to effectively contribute their outgoing bandwidth and leads to the scalability of mesh-based P2P streaming mechanisms. In practice, a group of peers in the overlay may exhibit a stronger (biased) internal connectivity within the group and weaker connectivity to peers outside the group, *i.e.*, a group of peers form a cluster. Such clusters in the overlay can form for various reasons including the localization of the overlay within an ISP, using regional bootstrap nodes coupled with establishing network-aware connections, and arrival of large number of peers in a short window of time (*i.e.*, flash crowd) . In particular, the localization of overlay connectivity within each edge ISP has received a great deal of attention in recent years [111]. The assumption in these studies is that localization does not adversely affect the performance of P2P applications. Thus, the main focus on these efforts is on providing an interface between ISP and P2P applications to facilitate the localization of connectivity within edge ISPs. In short, the effect of overlay localization on the performance of mesh-based P2P streaming of live content has not received much attention and is not well understood.

In this chapter, we present our ongoing work on understanding the effect of overlay clustering (or localization) on the performance of mesh-based P2P streaming mechanisms especially for delivery of live content. We leverage the idea of two-phase content delivery of *diffusion* and *swarming* in a certain class of mesh-based P2P streaming solutions as we have describe in Chapter III. The notion of two-phase content delivery clearly demonstrates the impact of overlay connectivity on the performance of content delivery.

8.1. Contributions & Design Objectives

Our goal in this work, is understanding and minimizing the effect of overlay clustering (or localization) on the performance of mesh-based P2P streaming mechanism without changing the block scheduling algorithm. The key questions that we want to address are as follows:

- How to detect any biased connectivity in the overlay that affects the performance of content delivery in mesh-based P2P streaming mechanisms?

- How to improve the performance of content delivery with minimum cost in terms of overhead, localization or stretch in the network?

Towards that we design a distributed Overlay Monitoring and Repair (OMR) mechanism that maintains proper connectivity of the overlay. As we discuss in Section 8.3., the key idea in OMR, is to use delivered quality to individual peers to identify and properly repair any major clustering in the overlay. More specifically, individual peers leverage the unavailability of a substream k as a signal for poor connectivity from subtree k. OMR employs a probabilistic approach to control the number of reacting peers and to increase the likelihood of reaction by properly-positioned peers without any coordination among participating peers. Reacting peers rewire the overlay by swapping their least useful parent with a parent in the subtree with poor connectivity. Our goal is to minimize the number of reactions for two reasons of minimizing the induced churn resulting from swapping parents and more importantly keeping the desired properties of the initial overlay (*e.g.*, localization) to the extent that is tolerable for mesh-based P2P streaming mechanisms. Thus, even the resulting rewired overlay by OMR mechanism, is significantly more localized/clustered than a random overlay mesh. This suggests that there is an opportunity for overlay localization without compromising the performance of mesh-based P2P streaming of live content.

8.2. Background

To demonstrate the effect of overlay connectivity on the performance of mesh-based P2P streaming, we use an organized view of a randomly connected and directed overlay that is introduced in Subsection 3.4.2. and shown in Figure 93. for clarity of discussion.

We consider a resource constraint setting where *(i)* source has limited bandwidth which is sufficient to send a live video stream without any redundancy (*i.e.*, a single copy of each block), and *(ii)* the demand for resources (aggregate incoming bandwidth) is equal to the available resources (aggregate outgoing bandwidth), *i.e.*, resource index is 1. We focus on a class of mesh-based P2P streaming mechanisms that employs a block scheduling scheme that is discussed and evaluated in Chapter III which prioritizes pulling of blocks with the largest timestamp [8, 127, 69, 138]. As we have shown in Chapter III the pattern of delivery for a single block over an organized overlay in such a block scheduling scheme consists of diffusion and swarming phases. Recall that the *diffusion connections* are the connections from peers in level n to peers in level $n + 1$ ($n<Depth$). Such connections are shown with straight edges in Figure 93.. Figure 93. also shows the three "diffusion subtrees" rooted at peer 1, 2 and 3 in an overlay. *Swarming connections* from a peer in level n to a peer in level m ($\leq n$) are shown with curly edges in Figure 93..

We call the collection of blocks that are delivered to a particular peer in level 1 and thus diffuse through the same diffusion subtree as a *substream* of the content.

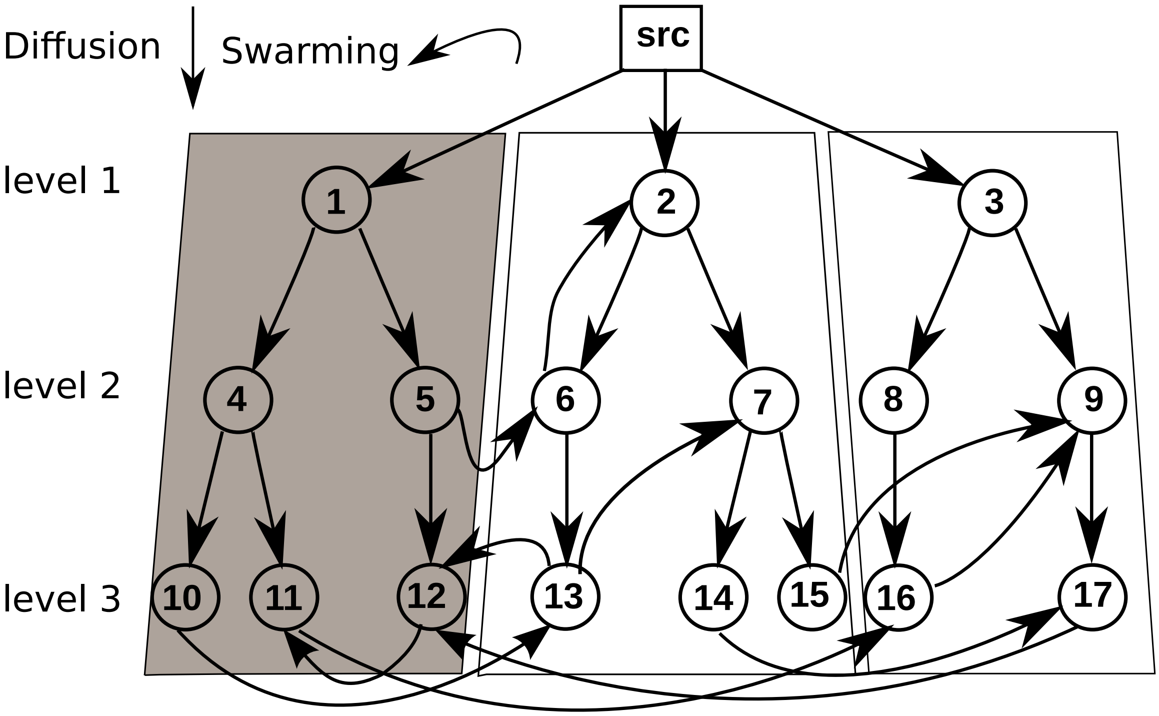


FIGURE 93.: Organized view of a random mesh overlay.

Since each block is sent only once, the content of different substreams are mutually exclusive. We identify these substreams by the id of the root of their corresponding subtree. For example, in Figure 93., the shaded subtree rooted at peer 1 has the $substream_1$ while the other subtrees rooted at peer 2 and 3 deliver $substream_2$ and $substream_3$, respectively.

8.2.1. Impact of Overlay Connectivity

The notion of diffusion and swarming phases for delivery of each substream (or block) enable us to relate buffer requirement and delivered quality for individual peers.Recall that the maximum number of required intervals for the diffusion of each substream equals to the *Depth* of its diffusion subtree. Moreover, the number of swarming intervals for a substream to a peer depends on the relative location of its swarming parents. Typically, a peer has one diffusion parent and several swarming parents. If the swarming parent of a peer is located in the diffusion subtree of the corresponding substream, then it only needs one extra interval to receive that substream. In general case, different swarming parents may not be located at the proper diffusion subtrees and thus require up to three swarming intervals in a randomly connected overlay which is derived mathematically and through simulation [8, 64] and discussed in Section 3.6.1.. For example, peer 7 in Figure 93., has the parent 13 in the same diffusion subtree. In this case, peer 7 can receive $substream_1$ through a longer path from parent 13.

Recall that, we can divide swarming connections into four categories based on the relative location of connected peers as discussed in Subsection 4.3.1. as follows: (i) C_{ld}: connecting peers at the bottom of two different diffusion subtrees (*e.g.*, connection from 10 to 13 in Figure 93.), *(ii)* C_{ls}: connecting peers at the bottom of the same diffusion subtree (*e.g.*, connection from 12 to 11 in Figure 93.), *(iii)* C_{id}: connecting a peer at one diffusion subtree (bottom or not) to an internal peer at a different diffusion subtree (*e.g.*, connection from 15 to 9 or 5 to 6), and (iv) C_{is}: connecting a peer at one diffusion subtree (bottom or not) to an internal peer at the same diffusion subtree (*e.g.*, connection from 13 to 7 or 6 to 2). Swarming connections that are between peers in different diffusion subtrees (*i.e.*, C_{ld} and C_{id}) are more useful for effective delivery of content. In particular, C_{ld} connections are most useful for swarming. These connections provide $substream_k$ for peer p at the bottom of diffusion subtree l. Since p is at the bottom of its own diffusion subtree (*i.e.*, its outgoing connections are not used for diffusion), it can effectively relay $substream_k$ to other peers in the same or different diffusion subtrees. Therefore, increasing C_{ld} swarming connections results in a large improvement in delivery of all substreams to individual peers within a small buffer size.

8.2.2. Effect of Overlay Clustering

In practice a group of peers may have a higher tendency to connect to each other and form clusters. Such a clustering effect might occur for various reasons. For example, ISP may provide an interface (such as P4P [111]) for its client peers to discover and connect to each other in order to reduce the number of external connections and thus limit the associated traffic. Peer discovery mechanisms that rely on local bootstrapping node and ping response from discovered peers are likely to exhibit similar clustering effect among close-by peers. Furthermore, a combination of flash-crowd event with certain pattern of peer arrival could also lead to cluster formation. We note that the dynamics of peer participation may reduce such a clustering event in some but not all scenarios.

Formation of clusters in the overlay causes a poor connectivity between different clusters that limits the flow of content among them. This may adversely affect the

Parameter	Description
$OutDeg_p$	Number of children of peer p.
$InDeg_p$	Number of parents of peer p.
$Depth_p$	Shortest depth of peer p across all subtrees.
Max_Depth_s	Maximum depth of diffusion subtree s.
$DiffSub_p$	The diffusion subtree of peer p.
$NumSub$	Total number of diffusion subtrees.
$Child(p,i)$	The ith children of peer p.
$Parent(p,i)$	The ith parent of peer p.
$SWParent(p,i)$	The ith swarming parent of peer p.
$Sub(p,s)$	is 1 when peer p receives substream s, o.w. it 0.

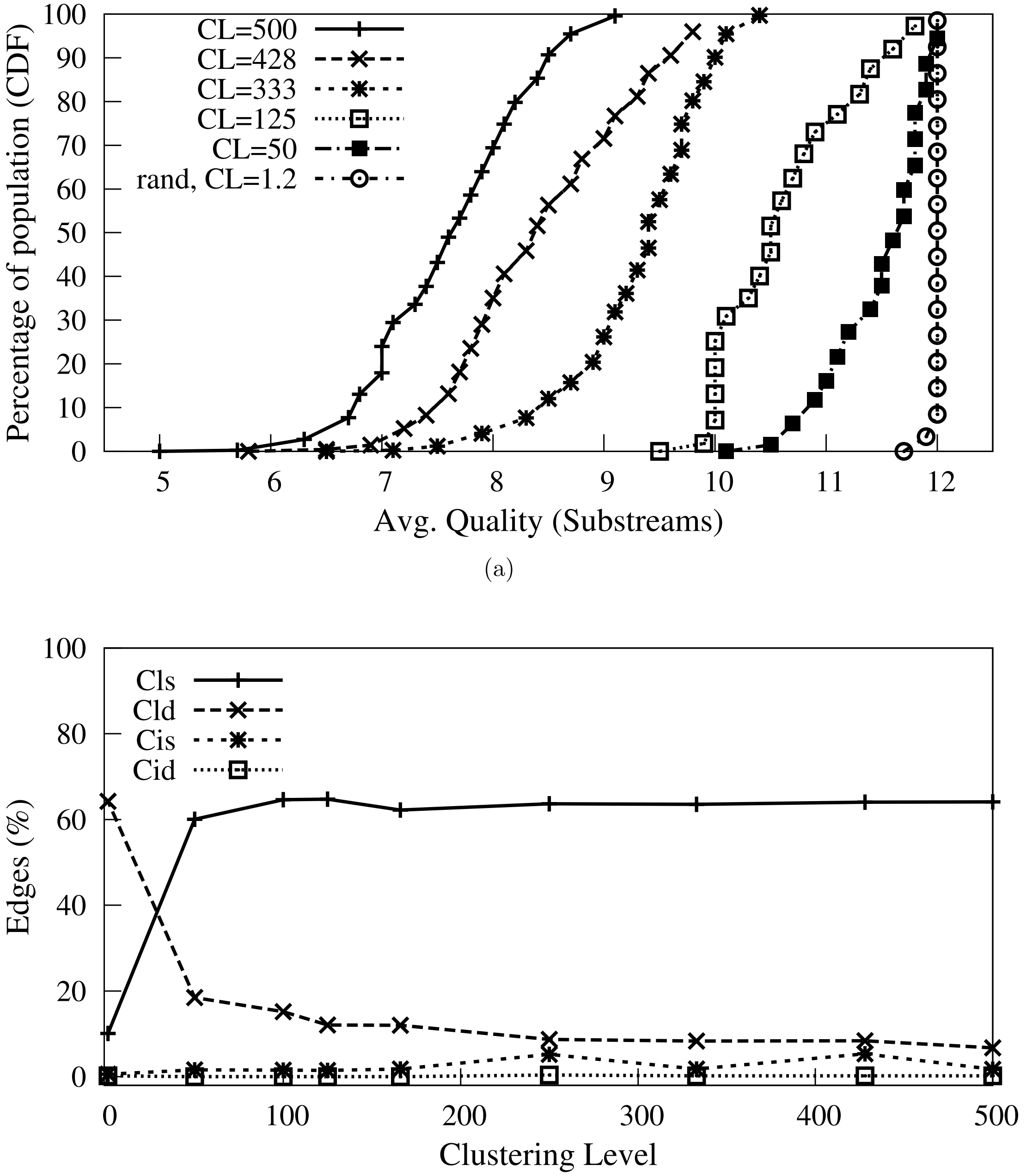
TABLE 13.: Summary of used parameters.

performance of content delivery. For example, if incoming external connections for a cluster do not pull mutually exclusive substreams, the full quality version of the stream does not reach that cluster and thus all peers in the cluster receive lower quality stream.

To further elaborate on the effect of overlay clustering on content delivery, we note that the diffusion subtrees and swarming connections in a clustered overlay are formed based on the relative distance of peers from the source as we described earlier. Peers in a particular cluster can be mapped to one or multiple diffusion subtrees depending on their external connections to other clusters and source. For example, if all peers in a cluster become part of a diffusion subtree, then the subtree should have many swarming connections within the subtree (of type C_{ls} or C_{is}), and very few swarming connections to other subtrees (of type C_{ld} or C_{id}). The limited number of swarming connections from $subtree_k$ to other subtrees could increase the number of required swarming intervals to receive $substream_k$ or could make $substream_k$ unreachable to other peers even with a large number of swarming intervals.

Alternatively, peers in a cluster could be mapped to different diffusion subtrees. In this case, the effect of biased connectivity on content delivery may vary depending on the relative location of peers in different subtrees. The swarming connections within a cluster connect corresponding regions of two subtrees at the cost of lower (or no) swarming connections to other diffusion subtrees. Such a focused inter-subtree swarming connections are not very useful because they may connect peers at the higher levels of two diffusion subtrees. In essence, any clustering in the overlay, decreases the number of C_{ld} connections between subtrees compared to the random overlay which adversely affects the performance of content delivery.

We use simulations to demonstrate the effect of overlay clustering on the content delivery. Consider 5000 homogeneous peers with $InDeg$ and $OutDeg$ of 12, that are grouped into 10 clusters (or ISPs). Source has sufficient bandwidth to deliver a single copy of each block (or substream), therefore, its $OutDeg$ is 12 which implied that the number of subtrees are 12. Figure 94.(a) depicts the distribution of delivered quality to individual peers for various level of clustering. We define the level of clustering CL based on the total number of incoming connections to all peers in a cluster divided by the number of external incoming connections to the cluster. Figure 94.(a) shows that the delivered quality significantly decreases with the level of clustering. Even with a relatively low level of clustering (*e.g.*, 50) the performance



(a)

(b)

FIGURE 94.: Effect of clustering: (a) Distribution of average delivered quality. (b) Percentage of various types of swarming connections.

is smaller than the random overlay which has the clustering level of 1.2. Figure 94.(b) depicts the percentage of various types of swarming connections as discussed in Subsection 8.2.1., as a function of the level of clustering. This figure reveals that with a higher level of clustering, C_{ld} connections decrease while C_{ls} connections increase. On the other hand, when we move towards a random overlay the number of good connections between subtrees (*i.e.*, C_{ld}) increases and those undesired connections within a subtree (*i.e.*, C_{ls}) decrease.

In summary, despite the subtle effect of clustering on overlay connectivity, its primarily relevant impact on content delivery is the limited (or no) availability of substream(s) that are associated with subtrees with limited swarming connectivity. Furthermore, the higher the number and quality (*i.e.*, type) of swarming connections from subtree k to subtree l are, the larger the number of peers in subtree l that can not receive $substream_k$ would be. In the next section, we leverage this point to devise a mechanism to detect poor connectivity in the overlay.

8.3. Overlay Monitoring & Repair

We propose a QoS mechanism, called OMR, to maintain the connectivity of the overlay such that swarming content delivery operates properly, and a large fraction of peers receives the desired quality. Such a QoS mechanism, requires a *detection* component that identifies any problem with overlay connectivity, and a *reaction* component that repairs the overlay connectivity by rewiring the minimum number of connections. The QoS mechanism should not require coordination among peers and should be light weight in order to scale to large groups. It should also minimize the number of changes in the connectivity of the overlay to limit the resulting dynamics in the overlay.

The main intuition in OMR is as follows: when a good block scheduling scheme is used, the limited availability of $substream_k$ in the diffusion subtree l indicates poor connectivity from subtree k to subtree l. Therefore, one can monitor the performance of content delivery (*i.e.*, delivered quality to individual peers) instead of connectivity in the overlay. This is an important distinction for two reasons: *(i)* identifying problems with overlay connectivity in a scalable fashion is expensive, and *(ii)* the effect of overlay connectivity on content delivery is rather subtle. We argue that the performance of content delivery is the only relevant metric since any type of clustering that does not affect content delivery, is not a concern.

Our proposed OMR mechanism is distributed and demand-driven by only leveraging the observed quality at each peer. Each peer p monitors its received substreams (*i.e.*, received quality) as well as the available substreams among its swarming parents (*i.e.*, available quality). When Peer p is missing at least one substream for a given period of time, it periodically invokes a *Reaction Algorithm* to determine whether it should react. If peer p is selected to react, it invokes a *Repair Algorithm* that implements the minimum number of changes in overlay connections to achieve the maximum improvement in the delivered quality.

Table 13. summarizes our notations we use throughout the chapter. We assume that each block carries a hop count that is set to zero and increases by each node that relays the block forward. This allows each peer to determine its shortest path from source (and its diffusion parent) as the overlay evolves. Source delivers only a single copy of each block and it tags individual blocks by their substream id based on the child peer (*i.e.*, root of the diffusion subtree) where the only copy of the block is delivered. The substream tag in blocks allows each peer to estimate the delivery of each substream and its availability among its swarming parents. Finally, each peer is aware of "the effective (or average) Max_Depth" of the overlay. This information can either be deducted from the hop count of received blocks or be estimated from the population and degree of participating peers.

8.3.1. Reaction Algorithm

A problem in overlay connectivity often affects delivered quality to a number of peers. The reaction algorithm is independently run by individual peers to determine which subset of affected peers should react to a drop in quality. More specifically, this algorithm tries to achieve two goals: *(i)* limiting the number of reacting peers, and *(ii)* increasing the probability of reaction by peers that are able to achieve the most improvement in overlay connectivity for a fixed number of rewiring operations due to their position in the overlay. The basic problem is very similar to selecting proper nodes to send a repair request in Scalable Reliable Multicast (SRM) [139].

Procedure 1 Reaction Algorithm

```
Require: $Sub(p, s) == 0$
$INPUT : p, s$ {Peer $p$ does not receive substream $s$}
$cont \leftarrow 0$, $pcont \leftarrow 0$
for $i = 1$ to $OutDeg_p$ do
    $c \leftarrow Child(p, i)$
    /* Compute the number of swarming children that do not have substream s */
    if $Sub(c, s) == 0$ and $Depth_c <= Depth_p$ then
        $cont \leftarrow cont + 1$
    end if
end for
for $i = 1$ to $InDeg_p$ do
    $q \leftarrow SWParent(p, i)$
    /* Compute the number of swarming parents in the same diffusion subtree that
    do not have substream s */
    if $Sub(q, s) == 0$ and $DiffSub_p == DiffSub_q$ then
        $pcont \leftarrow pcont + 1$
    end if
end for
$CF \leftarrow (1 + cont)/(1 + NumSub)$ {Probability of reaction is proportional to the
direct contribution of peer.}
$PF \leftarrow 1/(1 + pcont)$ {Inversely proportional to the number of swarming parents
that does not have this subtree and are in the same diffusion subtree as the peer
itself.}
$DF \leftarrow Depth_p/Max\_Depth_s$ {Lower depth peers should have a higher probability
of reaction}
$Prob\_React \leftarrow CF * PF * DF$
```

Thus, we adopt a similar "probabilistic approach" to control the number and the location of reacting peers. Each affected peer periodically estimates the probability of reacting to a missing substream. The pseudo code for reaction algorithm is presented in Procedure 1. The reaction probability for peer p to the absence of $substream_s$ depends on the following factors:

- *Contribution Factor (cf)*: the fraction of p's swarming children that does not receive $substream_s$. If p receives $substream_s$, it can relay it to all these swarming children that are missing s.

- *Parent Factor (pf)*: the fraction of p's swarming parents that are located in the same diffusion subtree and do not receive $substream_s$. These parents are experiencing the same problem and may react as well. Note that the swarming connection from these parents are of type C_{ls} and C_{is} that are less useful for content delivery.

- *Depth Factor (df)*: the relative value of p's depth to the overlay depth (Max_Depth). This is motivated by the fact that the outgoing connections of peers at lower levels (*i.e.*, higher relative depth) are mostly swarming connections.

The probability of reaction is biased towards peers with larger potential contribution, smaller parent factor and larger relative depth. A peer with the above qualities would be perfectly positioned to repair the overlay. The reaction probability is computed as $cf^{\alpha} * df^{\beta} * \frac{1}{pf^{\gamma}}$ where α, β and γ determine the contribution of the above factors on the responsiveness of OMR.

Probability of reaction for a peer p to subtree deficit of s, is proportional to the contribution of the peer. More specifically, if p has a large number of swarming children that do not have s, it is more probable to react. On the other hand, if p has a large number of swarming parents in its own diffusion subtree that do not receive

s, the probability of reaction for p should be decreased as those parents are likely to react. Moreover, the lower the depth of the peer, the higher the probability of reaction would be as peers at the bottom of the diffusion subtree could contribute the swarming content.

8.3.2. Repair Algorithm

Once the reaction algorithm triggers a reaction, it should identify a proper position nd, in the missing subtree s. The goal is to place a peer with higher contribution at a higher position in the missing subtree s. The algorithm for determining the new depth in the corresponding diffusion subtree s is presented in Procedure 2. The intuition behind this design choice is that when peer p has many swarming children which are missing subtree s, peer p will be their parent for subtree s. Thus to shorten the buffer size for peers, it is desirable to position peer p at a higher level in the diffusion subtree s.

To prevent from selecting the same depth by multiple reacting peers, we compute the new depth probabilistically. Towards this end, we form a probability function: $P(nd, cf) = 1 - (1 - cf)^{(nd-1)}$. The value of the function increases by both cf and nd. We start from potential new depth of 2 and at each step we decide to choose the value ND as the new depth with probability $P(ND, CF)$. We repeat this process until we decide on an ND or we reach $ND = Max\_Depth_s$.

Procedure 2 Repair Algorithm

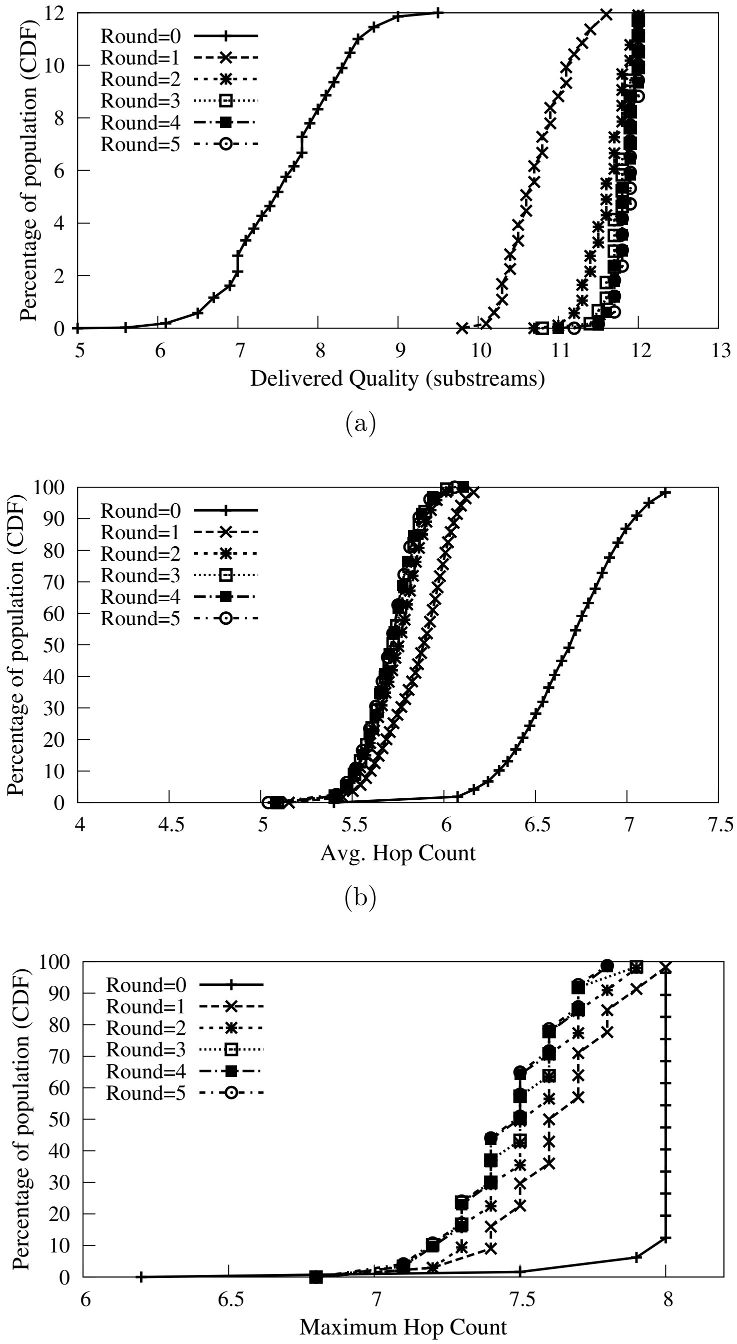
```
$INPUT$ : $p, s, CF$ {Peer $p$ does not have subtree $s$ and will establish a new connection}
$cont \leftarrow 0$
$pcont \leftarrow 0$
for $i = 2$ to $Max\_Min\_Depth_s$ do
    $pnewdepth \leftarrow 1 - pow((1 - CF), (i - 1))$
    $randn \leftarrow rand()$
    if $randn < pnewdepth || i == Max\_Depth_s$ then
        $return(i)$
    end if
end for
```

Upon computing the position in the *subtree*$_s$, peer p traverses the subtree in a depth-first search fashion by selecting a random child of each peer till it reaches the corresponding level.Finally, the reacting peer switches its current parent who does not provide any substream with this new parent. In case the new parent does not have an empty slot to admit a new child, it will disconnect one of its current children which has the least useful connection type (*i.e.*, of type C_{ls} and C_{is}). This new disconnected child will be swapped by peer p's previous parent.

8.4. Performance Evaluation

In this section, we present the preliminary evaluation of OMR using our event-driven simulator called *psim.* psim is an event-driven simulator that incorporates pairwise network delay between peers using King dataset [128]. The simulation is performed in multiple rounds where each round includes two steps: *(i)* P2P content delivery over the existing overlay and *(ii)* running the OMR mechanism at each peer which may cause rewiring of some connections. The buffer size that is set to $\log_{OutDeg}(N/Deg_{src})+3$. The delivered quality to individual peers is measured based on the number of delivered substreams. Clearly, if the connectivity of the overlay is inadequate, some peers may not receive all the required substreams and thus observe lower delivered quality. Round 0 starts with a "localized overlay" and the overlay is progressively rewired as a result of running OMR mechanism across individual peers in each round. In essence, each round can be viewed as the interval between two consecutive repair events. The simulation ends when the overlay does not experience any further change for a few rounds. For simplicity, we do not incorporate churn in our simulations.

Simulation Settings: We use the following default settings in our simulations: The required incoming degree to receive full quality stream is 12. The source bandwidth is equal to the stream rate, therefore source degree is also 12. Peers have homogeneous and symmetric bandwidth. Thus all peers are able to receive full quality stream and resource index is 1. Peer population is 5000 that is divided into 10 equal-sized clusters (or ISPs). The initial "localized overlay" has the highest level of clustering where each cluster has the minimum number of external connections, namely 12, and clusters form a randomly connected mesh. α, β and γ are set to 1, 2 and 2, respectively. The reported results for each simulation are averaged across ten runs with different random seeds.

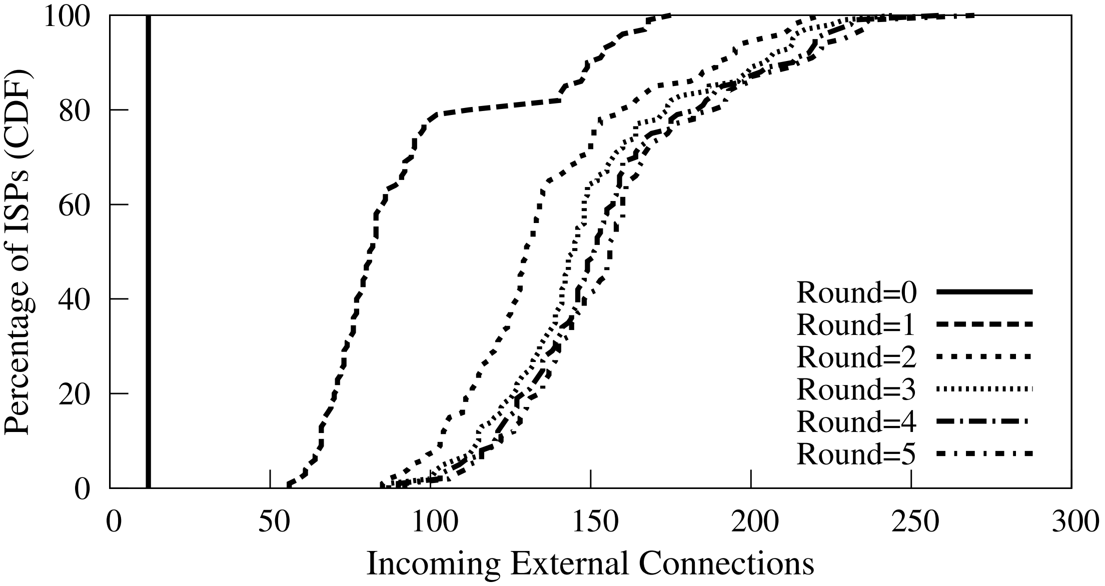


(a)

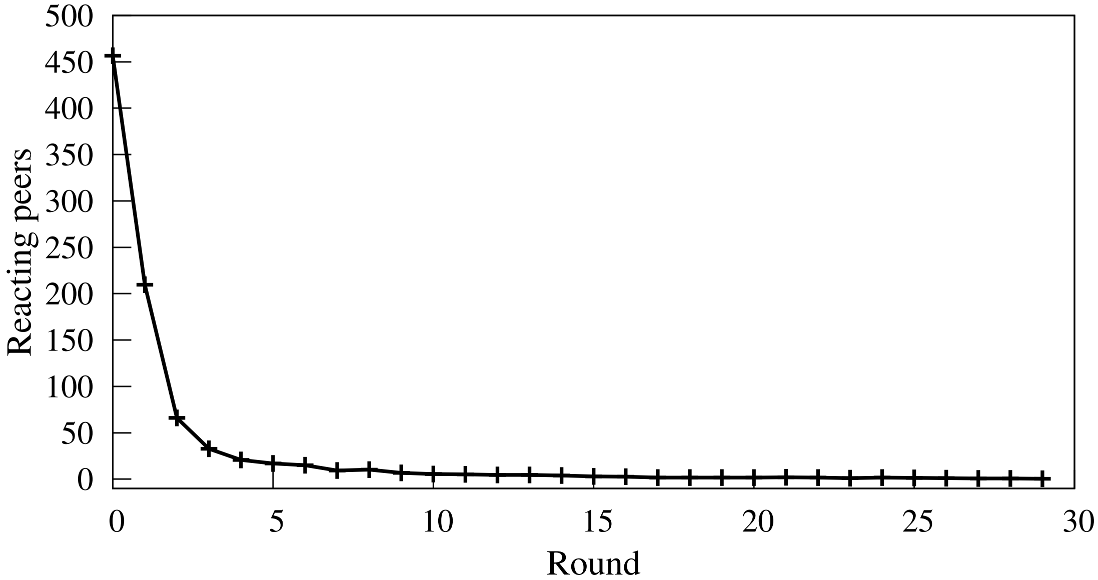
(b)

(c)

FIGURE 95.: Basic behavior of OMR: (a) Distribution of average delivered quality. (b) and (c) Distribution of average and maximum hop count across peers in various rounds, respectively.



(a)



(b)

FIGURE 96.: Basic behavior of OMR: (a) Distribution of incoming external connections to ISPs. (b) Percentage of reacting peers.

8.4.1. Benefits of OMR

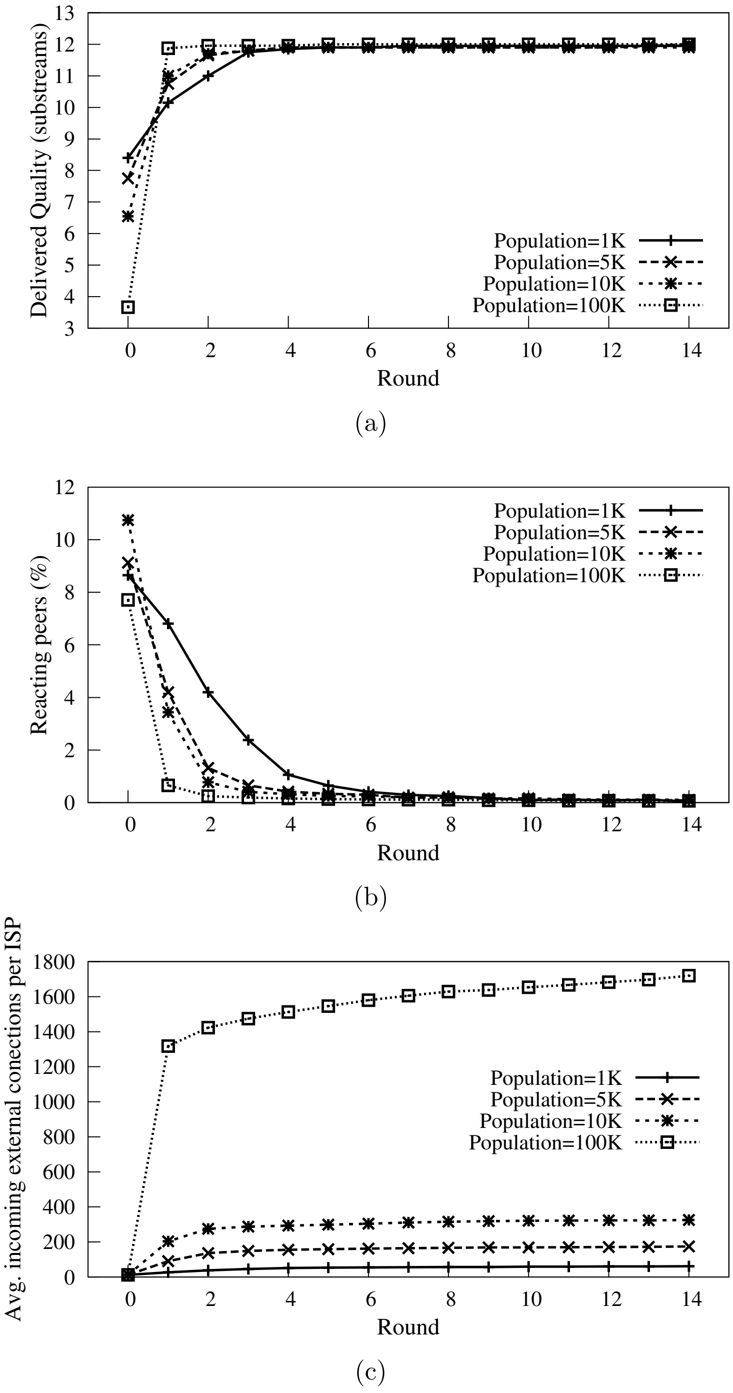
Each line of Figure 95.(a) depicts the CDF of delivered quality in separate rounds in our default scenario. In essence, this figure illustrates the incremental effect of OMR rewiring in each round on the delivered quality. This figure reveals that OMR leads to significant improvement in quality that occurs mostly during the first two rounds. Each line in Figure 95.(b) and 95.(c), shows the CDF of the average and maximum hop count for delivered blocks to individual peers in different rounds, respectively. These figures clearly demonstrates how OMR effectively tightens the connectivity and rapidly reduces average and maximum hop-count for delivery of blocks to all peers in the overlay within a couple of rounds. Given the population of peers and node degree, the buffer size at each peer is set to (Max_Depth+3 or) 8 intervals in these simulations. This explains why the hop-count is capped at 8 intervals.

Figure 96.(a) illustrates the effect of OMR on the increasing connectivity among clusters by showing the distribution of cumulative number of external connections across all clusters (ISPs) during the initial rounds. In round 0, all clusters in the localized overlay have exactly 12 external edges. After one round, OMR results in 50 to 170 external connections for different clusters. After two rounds, the number of external connections increases to the range 90 to 220. The number of external connections quickly stabilizes and does not exhibit any significant increase during the following rounds. In fact, no more rewiring occurs after the 10th round. The relative diversity in the number of established external connections among clusters caused by their initial external connectivity and relative position from the source. For example, a cluster that has two incoming connections from source contains two diffusion subtrees and establishes fewer external connections. It is worth noting that the resulting rewired overlay is still much more clustered than a comparable random overlay.

Finally, Figure 96.(b) depicts the number of reacting peers (or the number of rewired operations) in each round. This figure reveals that roughly 9% of edges (*i.e.*, 450 edges) are rewired in the first round. However, the number of rewired edges rapidly drops in the following rounds. This figure shows that OMR achieves a short response time by aggressively rewiring the overlay during a few rounds at the cost of rewiring a relatively large percentage (9%) of edges in a single round. Since the number of rewired edges in one round could be a concern, the reaction algorithm can be tuned through the configuration parameters α, β and γ introduced in Section 8.3., to reduce the number of reacting peers. This controls the number of rewiring events in one round but is likely to increase the response time. We are currently exploring this issue.

8.4.2. Scalability of OMR

To examine the scalability of the OMR, we evaluate its behavior on overlays with 1K, 5K, 10K and 100K peers. Each line in Figure 97.(a) shows the average



(a)

(b)

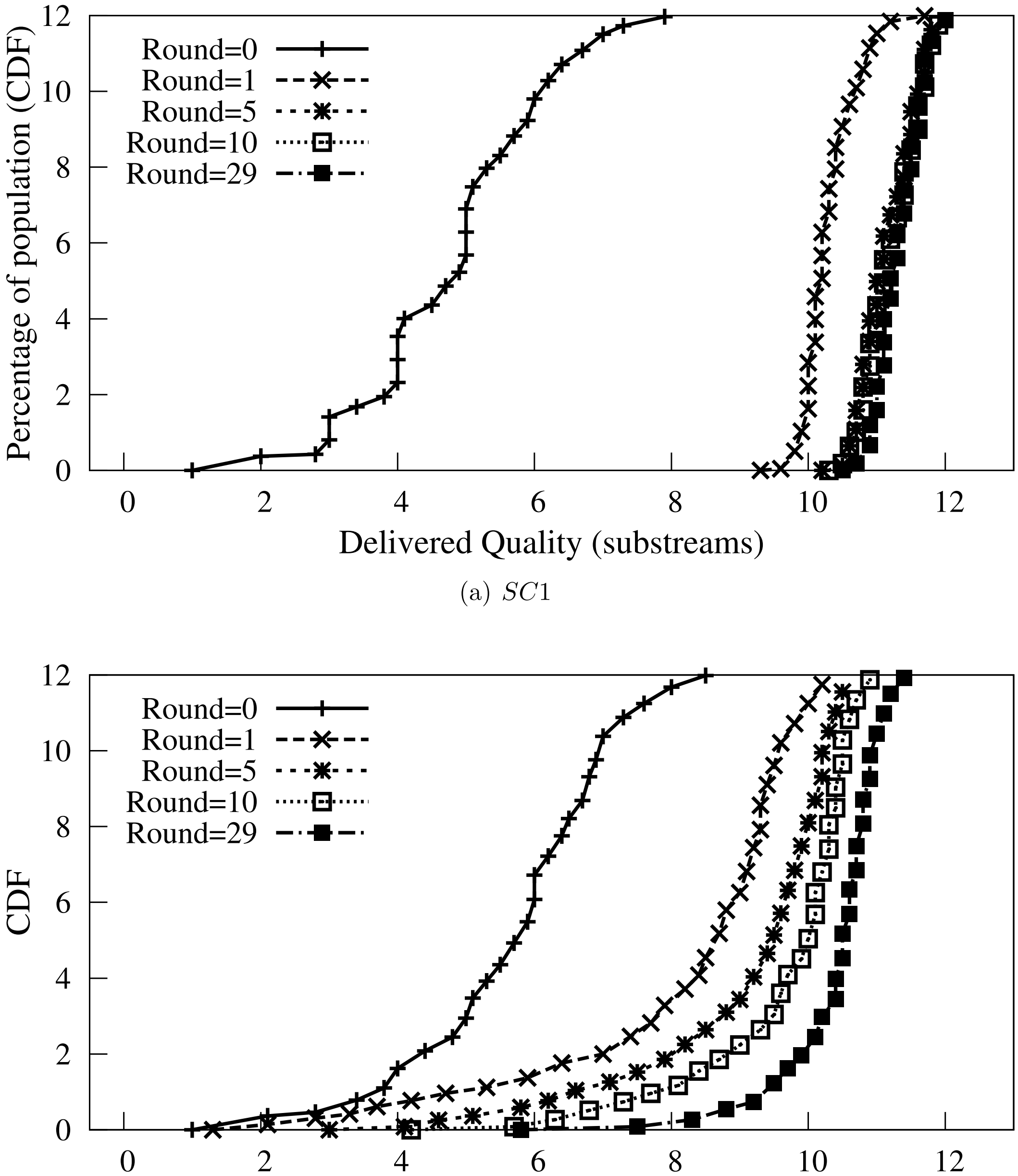
(c)

FIGURE 97.: Effect of peer population: (a) Distribution of delivered quality across various rounds. (b) Percentage of reacting peers across various rounds. (c) Average number of external incoming connections per ISPs for various rounds.

delivered quality to peers in an overlay with certain population as a function of time (rounds). This figure reveals that the average delivered quality rapidly improves during the first couple of rounds and reaches the full quality regardless of peer population. Figure 97.(a) also reveals that the delivered quality in large overlays is initially lower than smaller overlays. This is due to their lower ratio of external to internal connections for each cluster in larger overlays. Note that the number of external connections for the localized overlay is fixed at 12. To generate larger localized overlays, we increase the number of peers and edges within each clusters without increasing the number of external connections. This in turn increases the level of clustering in the overlay and adversely affects the performance of content delivery. Figure 97.(a) indicates that the time to reach the maximum quality is shorter in larger overlays despite their higher quality deficit in round zero. The reason is that the lower delivered quality in larger overlays triggers a larger number of peers to react initially which leads to a rather larger improvement in the delivered quality.

To quantify the reaction of OMR mechanism with a different population of peers, Figure 97.(b) depicts the percentage of reacting peers as a function of rounds for overlays with different populations. This figure shows that roughly the same fraction of peers react in the first round regardless of peer population. However, the absolute number of reacting peers in a larger overlay is proportionally larger. After the significant increase in the delivered quality the first round, the percentage of reacting peers in the following rounds rapidly drops to zero for all overlays with any population.

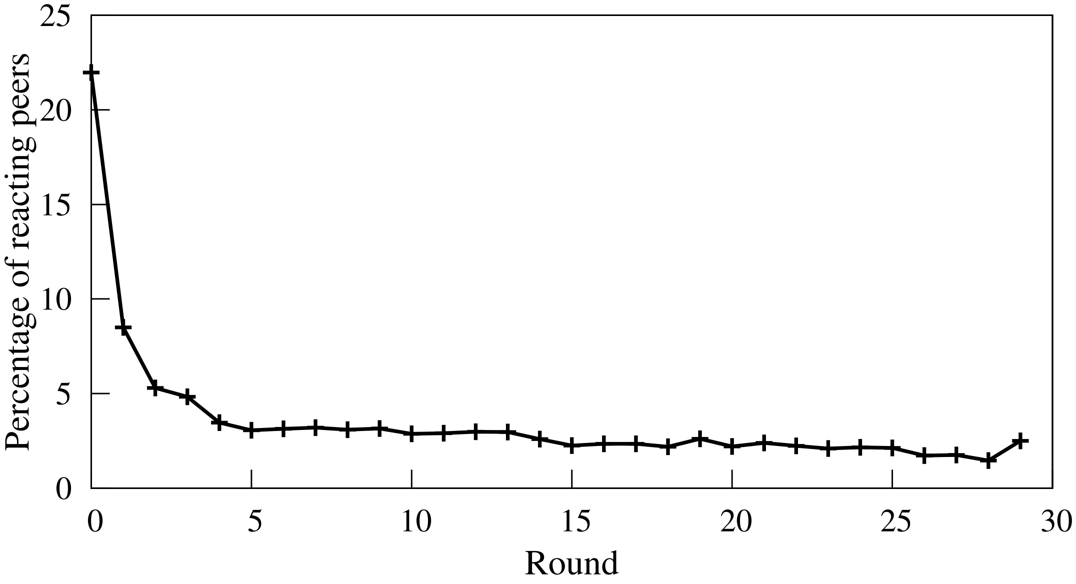
To characterize the effect of rewiring on the connectivity among ISPs, Figure 97.(c) shows the average number of external incoming connections per ISP as a function of rounds. This figure shows that the number of external connections rapidly increases to a certain level (*i.e.*, 80, 200, 350, and 1700) and further stabilizes. We note that the swarming intervals in a clustered overlay could be larger than 3 intervals due to the limited external connectivity among clusters. The higher the level of clustering, the larger the required number of swarming intervals. Since we fix the number of swarming intervals in the simulations, a larger number of peers receive a low quality in larger overlays which triggers the repair by a larger number of peers. These repairs result in establishing new external connections between clusters. The external connections serve as shortcuts in the overlay, reduce the number of swarming intervals, and thus increase the delivered quality which prevents further reaction by other peers. As part of our future work, we plan to mathematically derive the minimum required external connections for each cluster to ensure high delivered quality to individual peers in various scenarios. This reveals any potential gap between the required and the observed external connectivity in these scenarios.



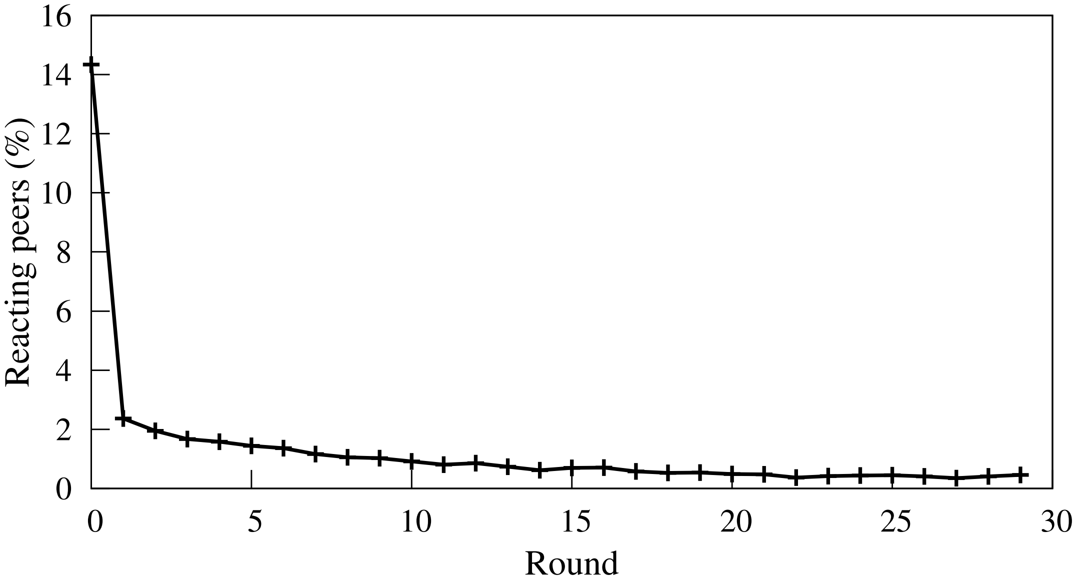
(a) $SC1$

(b) $SC2$

FIGURE 98.: Effect of bandwidth heterogeneity: (a) and (b) Distribution of delivered quality across high-bandwidth peers in $SC1$ and $SC2$, respectively.



(a) $SC1$



(b) $SC2$

FIGURE 99.: Effect of bandwidth heterogeneity: (a) and (b) Percentage of reacting peers in $SC1$ and $SC2$, respectively.

8.4.3. Peer Bandwidth Heterogeneity

We now examine the performance of OMR mechanism in more realistic scenarios with two groups of peers with high and low symmetric access link bandwidths. We assume that content is encoded with MDC and high bandwidth peers are able to receive the full stream quality (all 12 substreams) while low bandwidth peers can only receive 50% of the quality (6 substreams). The incoming and outgoing degree of high and low bandwidth peers are 12 and 6, respectively. We consider two scenarios of $SC1$ and $SC2$ with 5000 peers where 80% and 20% of peers are high bandwidth and all other peers are low bandwidth, respectively. Low bandwidth peers require a smaller number of substreams and are less sensitive to unavailability of a particular substream. Therefore, we primarily focus on the delivered quality to high bandwidth peers.

Figures 98.(a) and 98.(b) depict the distribution of delivered quality to high bandwidth peers in different rounds of applying OMR mechanism to both scenarios. The distribution of delivered quality in both scenarios are rather similar in round 0, and OMR mechanism result in significant improvement in delivered quality during the first two rounds. However, the distribution of the delivered quality in $SC1$ (with 80% high bandwidth peers) quickly becomes uniform after a couple of rounds, while it always remains relatively skewed in $SC2$ with lower median for delivered quality. The observed behavior in $SC2$ can be explained as follows: the large percentage of low bandwidth peers in $SC2$ increases the likelihood that a high bandwidth peer p is connected to several low bandwidth peers as a parent or child in the overlay. Assuming peer p misses a substream s and some of its low bandwidth parents/children reach their target quality and do not require substream s.This leads to the decrease of the value of contribution and parent factors in the reaction algorithm for the high bandwidth peer p. Thus, the probability of reaction by some high bandwidth peers decreases despite the deficit in their quality in $SC2$.

Figures 99.(a) and 99.(b) depict the percentage of reacting peers (or rewiring events) in the above two scenarios. We observe that the response time in these heterogeneous scenarios is longer than the homogeneous scenario examined in earlier subsections. This is due to the subtle interaction of two factors *(i)* the number of low bandwidth parent/children for each high bandwidth peer, and *(ii)* whether these low bandwidth peers require the same missing substream or not. Figures 99.(a) reveals that the number of reacting peers appears to stabilize at a low rate but does not reach zero. This persistent residual reaction occurs when a particular rewiring event replaces one low bandwidth parent with another low bandwidth parent. This could ensure availability of a particular substream at the cost of losing another substream. As part of our future work, we plan to examine the above issues that arise in the behavior of OMR over heterogeneous overlays.

8.5. Summary

In this chapter, we presented our ongoing work on overlay monitoring and repair (OMR) mechanism for mesh-based P2P streaming of live content. OMR leverages the availability of content at individual peers to detect poor connectivity between different regions of the overlay due to regional clustering or localization. OMR uses a probabilistic approach to ensure an adequate but minimum number of affected peers with proper position in the overlay react to poor connectivity. Reacting peers carefully identify proper counterparts in the overlay and swap specific parents to improve the connectivity of the overlay with the minimum number of rewiring operation.

Our preliminary simulations results demonstrated that OMR is able to achieve its design goal, and revealed a few more issues that require further investigation. As part of our future work, we plan to investigate the following issues on the behavior of OMR mechanism in more detail: *(i)* analytically derive the reaction function based on the minimum total number of required connections between various subtrees, *(ii)* investigate the effect of various mappings of clusters to subtrees on the minimum number of required connections between clusters and subtrees that we have derived mathematically, *(i)* examine the behavior of OMR in other scenarios and looking at the effect of churn and peer bandwidth heterogeneity, *(ii)* evaluate the effect of reaction algorithm and its damping factor on controlling the trade off between responsiveness and aggressiveness of the OMR mechanism.

CHAPTER IX

CONCLUSION

Scalable delivery of live video through P2P network is challenging due to the nature of the content being delivered, lack of any network or infrastructure support (*i.e.*, pool of dedicated servers or IP Multicast) and relying on the resource contribution of unreliable ordinary Internet users with diverse capabilities in terms of contributed resources and processing power. My dissertation focuses on the design, evaluation and comparisons of live P2P streaming mechanisms in realistic scenarios. We proposed the design of a novel mesh-based P2P streaming mechanism for live content called, PRIME. Further, we compared the performance of PRIME, as a representative mesh-based protocol, to the CoopNet protocol that adopts a tree-based approach for delivery of live streaming. In seeking to comparing various mesh-based solutions, we proposed a new performance evaluation methodology that assisted us in performing such a comparison. Utilizing the proposed methodology, we systematically compared the performance of mesh-based solutions with respect to a wide range of design and environmental parameters. From a more practical perspective, we tackle two important challenges that current live P2P streaming mechanisms are facing, namely, resource management and P2P traffic localization.

This dissertation provides important insights into the design and evaluation of live P2P streaming mechanisms in realistic environments. Furthermore, my dissertation plays a key role in the systematic understanding of the behavior of various approaches to live P2P streaming mechanisms. In the follow subsections, I recap the main contributions of this dissertation and sketch my future plans.

9.1. Contributions

9.1.1. Design and Evaluation of Live P2P Streaming

We designed and systematically evaluated a novel mesh-based live P2P streaming mechanism called PRIME. Through a performance driven approach, initially we identified two main performance bottlenecks, namely bandwidth and content bottlenecks. To overcome these bottlenecks, we have shown that the global pattern of content delivery should have two phases, diffusion and swarming. Further, we derived the required block scheduling scheme employed by individual peers whose collective behavior across all peers leads to the desired pattern of delivery. Our proposed two-phase model for content delivery reveals the impact of overlay connectivity and source behavior of the content delivery. Moreover, it captures the fundamental limitations of the system by demonstrating the relation between population, connectivity and buffer requirement at individual peers. Through a detailed performance evaluation of PRIME under realistic and diverse environments, we carefully examine the effect of different parameters. We have shown that there is a sweet range of peer degree over which mesh-based live P2P streaming exhibits a good performance. Note that such findings are missing from previous works as most of the evaluations were performed through session-level simulations which fail to model packet-level dynamics and the impacts of congestion.

9.1.2. Performance Comparison of Live P2P Streaming Approaches

We systematically compared the performance of the two basic approaches to live P2P streaming, namely, tree- and mesh-based. We identified striking similarities and differences between these two approaches and evaluated their performance through packet-level and session-level simulations to capture both the effect of packet dynamics and dynamics of peer participation on these approaches. We have shown the superiority of mesh-based approach in most of the scenarios despite its additional signaling overhead. In essence, we have shown that, due to the static mapping of content to parents, the tree-based approach suffers in dynamics scenarios. The block scheduling scheme employed in the mesh-based approach overcomes such a problem by mapping the blocks to parents for a short period of time and adjusting the mapping according to network conditions.

9.1.3. Comparison Study of Mesh-based Live P2P Streaming Solutions

Towards the objective of comparing various mesh-based P2P streaming solutions, we proposed a new performance evaluation methodology for live P2P streaming mechanisms. Our methodology introduces a set of performance metrics that can effectively capture the performance of content delivery on live P2P streaming mechanisms. Further, we derived the minimum requirement for each metric in a mesh-based live P2P streaming mechanisms with good performance, and then offered a signature for performance evaluation. Leveraging our proposed methodology, we systematically evaluated the performance of commonly used mesh-based live P2P streaming solutions under realistic settings. We believe that our proposed methodology offers a simple and yet powerful approach to meaningfully evaluate mesh-based P2P streaming mechanisms. Moreover, our comparison study revealed that any poorly designed mesh-based P2P streaming mechanism can achieve good performance by adding the necessary amount of proper resources, namely source and/or peer bandwidth.

9.1.4. Resource Management in Live P2P Streaming

To address resource management and fairness issues in live P2P streaming mechanisms, we designed and proposed a mechanism for managing resources in highly dynamic P2P systems. Our mechanisms provides incentives for peers to contribute their resources in order to improve their delivered quality in a distributed fashion. Our mechanism allows high bandwidth peers to compensate for the resource-poor peers by contributing more bandwidth to the system and proportionally receiving better streaming quality. In other solutions based on a bit-for-bit fair sharing mechanism, peers with very high upload capacity cannot contribute all of their resources, while peers with low upload capacity are prevented from receiving more than their upload capacities. Through extensive simulations we have shown the effectiveness of our mechanism to allow peers receive different levels of quality based on the overall instantaneous available upload bandwidth in the system as well as the amount of their contribution.

9.1.5. P2P Traffic Localization

Localization or limiting inter-ISP P2P traffic significantly affects the performance of mesh-based live P2P streaming applications. This new and fundamental finding has been missing from previous work on P2P traffic localization which focused on how to find nearby peers to build a localized overlay. In this dissertation, we investigated the impact of localization and, through simulations and analysis showed that by increasing the localization the performance degrades. To achieve ISP-friendliness, we adopted two approaches, namely changing the block scheduling scheme, and revising the overlay connectivity without changing the block scheduling.

In the first approach, we proposed a novel two-tier overlay-aware block scheduling scheme, called OLIVES. OLIVES' design is performance-driven, meaning that, we identified fundamental limiting factors affecting the performance of basic mesh-based live P2P streaming mechanisms in fully localized overlays. OLIVES achieves good performance over a wide range of realistic scenarios while maximizing traffic localization. On the negative side, OLIVES significantly increased the required buffer size; we have proposed a method to compensate for the extra required buffer. We believe the ideas in OLIVES can be deployed not only in fully localized overlays but also in any clustered overlays with limited connectivity between clusters.

Adopting the second approach, *i.e.*, revising the overlay connectivity, we proposed our on-going work on an overlay monitoring and repair mechanism, called OMR, with the goal of keeping the overlay as localized as possible without degrading the performance of content delivery. OMR uses a probabilistic approach to ensure an adequate but minimum number of peers with poor quality react to revise the overlay in an efficient fashion. Through simulations we demonstrated that OMR can achieve its design goals and improve the performance of mesh-based live P2P streaming mechanisms. Compared to OLIVES, OMR achieves less localization; however, its simple scheduling scheme, smaller buffer size and non-reliance on any infrastructure make it suitable for non-commercial P2P streaming applications that do not generate a huge amount of P2P traffic.

9.2. Future Works

In this section, I present some research problems for future investigation. My future work can be divided into two broad areas: *(i)* problems that are related to the topic of P2P streaming which are motivated by my dissertation, and *(ii)* problems that are related to the interaction between P2P and other contexts, namely online social networks and AS-level underlays. In what follows I will briefly describe each of these research problems. The first three subsections belong to the first area of my interest and the last two subsections are the topics related to the interaction between P2P and other contexts.

9.2.1. Stochastic Fluid Model of Live P2P Streaming

A number of theoretical studies of live P2P streaming systems focus on analysis of the mesh-based approach [127, 140, 97, 141, 130, 142, 143, 144]. However, most of these works rely on unrealistic assumptions such as fully provisional systems, complete overlays, or focus on bandwidth as the only constraint on delivered quality. These unrealistic assumptions prevent the model from capturing the accurate behavior of live P2P streaming applications. Another important issue is the choice of modeled metric. Most of the existing works have analyzed the delivered quality to individual peers. Clearly, the delivered quality reflects the performance of peers but only at the surface level. In order to go deeper and understand the temporal performance of peers in the P2P streaming contexts, one should analyze a metric that captures the evolution of buffer state at each peer. Towards that goal, I have started to analytically capture the performance of mesh-based live P2P streaming mechanisms by proposing a stochastic fluid model under realistic resource-constraint settings in which the peer degree is bounded while both bandwidth and content availability are considered as the two performance bottlenecks. My goal is to analyze the content availability or the buffer state at individual peers which further can lead to the observed performance by each peer.

9.2.2. Multi-Channel P2P Streaming

An interesting extension of this dissertation is to support multi-channel live P2P streaming applications. I would like to explore this topic with the goal of improving user experience in surfing or switching between channels. Locating peers with excess bandwidth in other channels, sharing resources (*i.e.*, bandwidth) between channels, minimizing the switching delay and coping with the additional dynamics due to switching channels are among the challenges in this context. One method to assist smooth channel switching is predicting users' behavior and establishing back-up connections with peers in other channels that the user is more likely to switch to. Such a prediction can be made based on history of the particular user's behavior or through studies on the average behavior of users watching TV channels.

9.2.3. Incentive Mechanisms in Live P2P Streaming

The contribution-aware mechanism that we have proposed in Chapter VI relies on cooperative peers that correctly report the amount of resources they are willing to contribute. However, in practice, some misbehaving users may report contributions larger than their actual contributions. In such a setting, two approaches can be adopted as follows: *(i)* The incentive mechanism can adapt the instantaneous direct reciprocity or a tit-for-tat approach, by allowing only bi-directional connections between peers and directly monitoring the amount of contributed resources of the direct neighbor, or *(ii)* There can be an additional monitoring mechanism in place to verify the faithfulness of peers in the system. As future work, I would like to explore these approaches in the context of non-cooperative live P2P streaming and explore how the tit-for-tat approach can be adopted in live P2P streaming mechanisms.

9.2.4. Social-aware P2P Systems

Joining a P2P streaming session or downloading a file is based on users' interests. Knowing users' interests a priori can provide opportunities for efficient neighbor discovery or P2P caching. Users sharing common social properties such as being friends, part of the same community, group or region, are more likely to have overlapping interests. Moreover, such interest can be affected by being informed about the availability of a certain video or file. Online Social Networks (OSNs) such as MySpace and Facebook consist of hundreds of millions of users. People that have a friendship relation in their actual life, establish friendship connections in OSNs. Moreover, users in OSNs, join various groups and communities based on their passions. Thus, information from OSNs can be leveraged to predict users' interests. For example, when a user posts a video on Facebook, some of her friends may watch that particular video. Users in the same political or sports group, may want to watch a video on a particular political debate or a live football event. Such information can be harvested and then utilized in P2P systems for efficient peer discovery (*i.e.*, routed towards friends first) or P2P caching (*i.e.*, a more active user caches a video). Design of such a social-aware P2P system requires studying and analyzing the social behavior of users in OSNs towards the prediction of their interests.

9.2.5. Interactions Between P2P Overlay and AS-level Underlay

There are several open issues to explore in the context of the interaction between the P2P overlay and the AS-level underlay. The huge volume and unique patterns of traffic related to P2P applications can both affect the policies imposed by ISPs in routing the traffic and the underlying link utilization. Consequently, these two factors impact the P2P application design and the connectivity between ISPs. From the applications point of view, ISP-friendly P2P applications can be designed adopting various approaches as described in Chapter VII and further the proposed approaches can be compared with each other. On the other hand, from the ISPs point of view, the impact of P2P applications on the evolution of AS-level connectivity can be explored. In this regard, an interesting research question is to examine the global and local impact of traffic generated by content delivery applications on ISPs in terms of cost and link utilization. Such applications can be streaming or file sharing and adapting different architectures for content delivery such as client/server, CDN, random P2P or localized P2P.

BIBLIOGRAPHY

[1] N. Magharei, A. H. Rasti, D. Stutzbach, and R. Rejaie, “Peer-to-Peer receiver-driven mesh-based streaming,” in *Proc. ACM SIGCOMM, Poster Session*, 2005.

[2] N. Magharei and R. Rejaie, “Understanding Mesh-based Peer-to-Peer Streaming,” in *Proc. ACM NOSSDAV*, 2006.

[3] N. Magharei, Y. Guo, and R. Rejaie, “Issues in Offering Live P2P Streaming Service to Residential Users,” in *Proc. IEEE CCNC*, 2007.

[4] N. Magharei and R. Rejaie, “Prime: Peer-to-Peer receiver-driven mesh-based streaming,” in *Proc. IEEE INFOCOM*, 2007.

[5] N. Magharei, R. Rejaie, and Y. Guo, “Mesh or Multiple-Tree: A Comparative Study of P2P Live Streaming Services,” in *Proc. IEEE INFOCOM*, 2007.

[6] N. Magharei and R. Rejaie, “ISP-friendly P2P streaming,” *IEEE MULTIMEDIA COMMUNICATIONS TECHNICAL COMMITTEE E-Letter*, 2009.

[7] N. Magharei and R. Rejaie, “Overlay monitoring and repair in swarm-based Peer-to-Peer streaming,” in *Proc. ACM NOSSDAV*, 2009.

[8] N. Magharei and R. Rejaie, “PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming,” *IEEE/ACM Transactions on Networking*, 2009.

[9] N. Magharei, R. Rejaie, and Y. Guo, “Incorporating contribution-awareness into mesh-based Peer-to-Peer streaming services,” *Peer-to-Peer Networking and Applications*, 2010.

[10] J. Kurose and K. Ross, *Computer Networking, A top-Down Approach Featuring the Internet*. Addison Wesley, 2002.

[11] Y. Guo, K. Suh, J. Kurose, and D. Towsley, “A Peer-to-Peer on-demand streaming service and its performance,” in *Proc. IEEE ICME*, 2003.

[12] K. Hua and S. Sheu, “Skyscraper broadcasting: A hybrid broadcasting scheme for metropolitan video on demand systems,” in *Proc. ACM SIGCOMM*, 1997.

[13] S. Sheu, K. Hua, and W. Tavanapong, “Chaining: A generalized batching technique for video-on-demand systems,” in *Proc. International Conference on Multimedia Computing and Systems*, 1997.

[14] S. Ramesh, I. Rhee, and K. Guo, “Multicast with cache (mcache): An adaptive zero-delay video-on-demand service,” in *Proc. IEEE INFOCOM*, 2001.

[15] J.-F. Paris, S. Carter, and D. Long, “A hybrid broadcasting protocol for video on demand,” in *Proc. Multimedia Computing and Networking*, 1999.

[16] D. A. Tran, K. A. Hua, and T. Do, “Zigzag: An efficient Peer-to-Peer scheme for media streaming,” in *Proc. IEEE INFOCOM*, 2003.

[17] “www.vseelab.com.”

[18] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, “A case for end-system multicast,” *IEEE Journal of Selected Areas in Communication*, 2002.

[19] P. Francis, “Yoid: Extending the multicast internet architecture,” in *Proc. White Paper, http://www.icir.org/yoid*, 1999.

[20] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, “Resilient Peer-to-Peer streaming,” in *Proc. IEEE ICNP*, 2003.

[21] V. Venkataraman, K. Yoshida, and P. Francis, “Chunkyspread: Heterogeneous Unstructured End System Multicast,” in *Proc. IEEE ICNP*, 2006.

[22] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, “Bullet: High bandwidth data dissemination using an overlay mesh,” in *Proc. ACM SOSP*, 2003.

[23] F. Wang, Y. Xiong, and J.Liu, “mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast,” in *Proc. IEEE ICDCS*, 2007.

[24] W. Yiu, X. Jin, and S. Chan, “Challenges and approaches in large-scale Peer-to-Peer media streaming,” *IEEE Multimedia*, 2007.

[25] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, “Resilient overlays using multicast,” *ACM/IEEE Transactions on Networking*, 2005.

[26] C. Yeo, B. Lee, and M. Er, “A survey of application level multicast techniques,” *Computer Communications*, 2004.

[27] M. Hosseini, D. Ahmed, , S. Shirmohammadi, and N. Georganas, “A survey of application-layer multicast protocols,” *Communications Surveys and Tutorials*, 2007.

[28] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "Almi: An application level multicast infrastructure," in *Proc. USNIX Symposium on Internet Technologies and Systems (USITS '01)*, 2001.

[29] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. ACM NOSSDAV*, 2002.

[30] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," in *Proc. IEEE INFOCOM*, 2002.

[31] D. Waitzman, C. Partridge, and S. Deering, "Distance vector multicast routing protocol," *RFC 1075, Internet Engineering Task Force*, 1988.

[32] S. Banerjee and B. Bhattacharjee, "A comparative study of application layer multicast protocols," University of Maryland, College Park, Tech. Rep., 2002.

[33] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, 2002.

[34] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group," in *Proc. ACM NOSSDAV*, 2002.

[35] V. Roca and A. El-Sayed, "A host-based multicast (hbm) solution for group communications," in *Proc. International Conference on Networking*, 2001.

[36] R. Douglas, "Np-completeness and degree restricted spanning trees," *Discrete Mathematics*, 1992.

[37] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a Peer-to-Peer network," Tech. Rep., 2001.

[38] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., "Overcast: Reliable multicasting with an overlay network," in *Proc. USENIX OSDI*, 2000.

[39] K. Sripanidkulchai, A. Ganjam, and B. Maggs, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proc. ACM SIGCOMM*, 2004.

[40] M. Guo and M. Ammar, "Scalable live video streaming to cooperative clients using time shifting and video patching," in *Proc. IEEE INFOCOM*, 2004.

[41] T. S. E. Ng, Y. Chu, S. G. Rao, K. Sripanidkulchai, and H. Zhang, "Measurement-based optimization techniques for bandwidth-demanding Peer-to-Peer systems," in *Proc. IEEE INFOCOM*, 2003.

[42] R. Carter and M. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proc. IEEE INFOCOM*, 1997.

[43] Z. Fei, S. Bhattacharjee, E. Zegura, and M. Ammar, "A novel server selection technique for improving the response time of a replicated service," in *Proc. IEEE INFOCOM*, 1998.

[44] K. Hanna, N. Natarajan, and B. Levine, "Evaluation of a novel two-step server selection metric," in *Proc. IEEE ICNP*, 2001.

[45] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proc. IEEE INFOCOM*, 2002.

[46] K. Lai and M. Baker, "Measuring link bandwidths using a deterministic model of packet delay," in *Proc. ACM SIGCOMM*, 2000.

[47] K. Lai and M. Baker, "Nettimer: a tool for measuring bottleneck link bandwidth," in *Proc. USENIX Symposium on Internet Technologies and Systems*, 2001.

[48] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relationship with tcp throughput," in *Proc. ACM SIGCOMM*, 2002.

[49] S. Savage, "Sting: A tcp-based network measurement tool," in *Proc. USENIX Symposium on Internet Technologies and Systems*, 1999.

[50] S. Shi, J. Turner, and M. Waldvogel, "Dimensioning server access bandwidth and multicast routing in overlay networks," in *Proc. ACM NOSSDAV*, 2001.

[51] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrustructure for real-time applications," in *Proc. IEEE INFOCOM*, 2003.

[52] Y. Zhong, S. Shirmohammadi, and A. E. Saddik, "Measurement of the effectiveness of application-layer multicasting," in *Proc. IEEE IMTC*, 2005.

[53] Z. Xu, C. Tang, S. Banerjee, and S. Lee, "Rita: Receiver initiated just-in-time tree adaptation for rich media distribution," in *Proc. ACM NOSSDAV*, 2003.

[54] Y. Chawathe, "Scattercast: An architecture for internet braodcast distribution as an infrusturucture service," *Ph.D. Thesis*, 2000.

[55] L. Mathy, R. Canonico, and D. Hutchison, "An overlay tree building control protocol," in *Proc. Workshop on Networked Group Communication*, 2001.

[56] M. Castro, P. Druschel, A.-M. Kermarrec, A. R. A. Nandi, and A. Singh, “Splitstream: High-bandwidth content distribution in a cooperative environment,” in *Proc. ACM SOSP*, 2003.

[57] G. Dn, V. Fodor, and I. Chatzidrossos, “On the performance of multiple-tree-based Peer-to-Peer live streaming,” in *Proc. IEEE INFOCOM, short paper*, 2007.

[58] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, “Scribe: A large-scale and decentralized application-level multicast infrastructure,” *IEEE Journal of Selected Areas in Communication*, 2002.

[59] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale Peer-to-Peer systems,” in *Proc. IFIP*, 2001.

[60] S. Xie, B. Li, G. Keung, and X. Zhang, “Large scale Peer-to-Peer live vide streaming: Theory and practice,” Tech. Rep., 2006.

[61] W. T. Ooi, “Dagster: Contributor aware end-host multicast for media streaming in heterogeneous environment,” in *Proc. Multimedia Computing and Networking*, 2005.

[62] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. Peterson, and R. Wang, “Overlay mesh construction using interleaved spanning trees,” in *Proc. IEEE INFOCOM*, 2004.

[63] J. Liang and K. Nahrstedt, “Dagstream: Locality aware and failure resilient Peer-to-Peer streaming,” in *Proc. Multimedia Computing and Networking*, 2006.

[64] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, “Coolstreaming: A data-driven overlay network for live media streaming,” in *Proc. IEEE INFOCOM*, 2005.

[65] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, “Chainsaw: Eliminating Trees from Overlay Multicast,” in *Proc. IPTPS*, 2005.

[66] F. Pianese, J. Keller, and E. W. Biersack, “Pulse, a flexible P2P live streaming system,” in *Proc. Global Internet Workshop*, 2006.

[67] M. Zhang, Y. Xiong, Q. Zhang, and S. Yang, “On the optimal scheduling for media streaming in data-driven overlay networks,” in *Proc. IEEE GLOBECOM*, 2006.

[68] A. Vlavianos, M. Iliofotou, and M. Faloutsos, “Bitos; enhancing bittorrent for supporting streaming applications,” in *Proc. Global Internet Workshop*, 2006.

[69] N. Magharei and R. Rejaie, “Dissecting the performance of Live Mesh-based P2P Streaming,” Tech. Rep. CIS-TR-07-05, 2007. [Online]. Available: http://mirage.cs.uoregon.edu/pub/tr07-05.pdf

[70] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, “Anysee: Scalable live streaming service based on inter-overlay optimization,” in *Proc. IEEE INFOCOM*, 2006.

[71] M. Zhang, J. Luo, L. Zhao, and S. Yang, “A Peer-to-Peer network for live media streaming using a push-pull approach,” in *Proc. ACM Multimedia*, 2007.

[72] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat, “Using random subsets to build scalable network services,” in *Proc. USENIX Symposium on Internet Technologies and Systems*, 2003.

[73] S. Saroiu, P. K. Gummadi, and S. D. Gribble, “Measurement study of Peer-to-Peer file system sharing,” in *Proc. Multimedia Computing and Networking*, 2002.

[74] E. Adar and B. Huberman, “Free riding on gnutella,” *First Monday*, 2000.

[75] B. Cohen, “Incentives build robustness in bittorrent,” in *Proc. Workshop on the Economics of Peer-to-Peer Systems*, 2003.

[76] P. Golle, K. Leyton-Brown, and I. Mironov, “Incentives for sharing in Peer-to-Peer networks,” in *Proc. ACM Electronic Commerce*, 2001.

[77] C. Buragohain, D. Agrawal, and S. Suri, “A game theoretic framework for incentives in P2P systems,” in *Proc. IEEE P2P*, 2003.

[78] D. Turner and K. Ross, “The lightweight currency protocol,” *Internet Draft*, 2003.

[79] M. Feldman, K. Lai, I. Stoica, and J. Chuang, “Scalable and robust incentive techniques for P2P networks,” in *Proc. ACM Electronic Commerce*, 2004.

[80] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster, “To share or not to share: An analysis of incentives to contribute in collaborative file sharing environments,” in *Proc. Workshop on Economics of Peer-to-Peer Systems*, 2003.

[81] T. Ngan, D. Wallach, and P. Druschel, “Enforcing fair sharing of Peer-to-Peer resources,” in *Proc. IEEE International Workshop on Peer-To-Peer Systems*, 2003.

[82] M. Feldman and J. Chuang, “Overcoming free-riding behavior in Peer-to-Peer systems,” in *Proc. ACM SIGecom Exchanges*, 2005.

[83] A. Habib and J. Chuang, "Incentive mechanism for Peer-to-Peer media streaming," in *Proc. IEEE International Workshop on Quality of Service*, 2004.

[84] A. Habib and J. Chuang, "Service differentiated peer selection: An incentive mechanism for Peer-to-Peer media streaming," *IEEE Transactions on Multimedia*, 2006.

[85] M. Gupta, P. Judge, and M. Ammar, "A reputation system for Peer-to-Peer networks," in *Proc. ACM NOSSDAV*, 2003.

[86] G. Tan and S. A. Jarvis, "A payment-based incentive and service differentiation mechanism for Peer-to-Peer streaming broadcast," in *Proc. IEEE International Workshop on Quality of Service*, 2006.

[87] K. Sripanidkulchai, A. Ganjam, and B. Maggs, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proc. ACM SIGCOMM*, 2004.

[88] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in a torrents lifetime," in *Proc. Passive Analysis and Measurement Workshop*, 2004.

[89] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent P2P file-sharing system: Measurements and analysis," in *Proc. Workshop on Peer-to-Peer Systems*, 2005.

[90] R. Thommes and M. Coates, "Bittorrent fairness: analysis and improvements," in *Proc. Internet, Telecom. and Signal Processing*, 2005.

[91] A. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving a bittorrent networks performance mechanisms," in *Proc. IEEE INFOCOM*, 2006.

[92] F. Pianese, D. Perino, J. Keller, and E. W. Biersack, "Pulse: an adaptive, incentive-based, unstructured P2P live streaming system," *IEEE Transaction on Multimedia*, 2007.

[93] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using layered video to provide incentives in P2P live streaming," in *Proc. workshop on Peer-to-Peer streaming and IP-TV*, 2007.

[94] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "Substream trading: Towards an open P2P live streaming system," in *Proc. IEEE ICNP*, 2008.

[95] Y. Sung, M. Bishop, and S. Rao, "Enabling Contribution Awareness in an Overlay Broadcasting System," in *Proc. ACM SIGCOMM*, 2006.

[96] Y. Chu, J. Chuang, and H. Zhang, "A case for taxation in Peer-to-Peer streaming broadcast," in *Proc. PINS, ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, 2004.

[97] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proc. IEEE INFOCOM*, 2007.

[98] S. Cheung, M. Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution," in *Proc. IEEE INFOCOM*, 96.

[99] T. Jiang, M. Ammar, and E. Zegura, "Inter-receiver fairness: a novel performance measure for multicast abr sessions," in *Proc. ACM SIGMETRICS*, 1998.

[100] Y. Chu, S. G. Rao, and H. Zhang, "A case for end-system multicast," in *Proc. ACM SIGMETRICS*, 2000.

[101] X. Li, M. Ammar, and S. Paul, "Layered video multicast with retransmission (lvmr): Evaluation of hierarchical rate control," in *Proc. IEEE INFOCOM*, 1998.

[102] S. McCanne, V. Jacobson, and M. Vettereli, "Receiver-driven layered multicast," in *Proc. ACM SIGCOMM*, 1996.

[103] J. Byers, M. Luby, and M. Mitzenmacher, "Fine-grained layered multicast," in *Proc. IEEE INFOCOM*, 2001.

[104] V. Goyal, J. Kovacevic, R. Arean, and M. Vetterli, "Multiple description transform coding of images," in *Proc. IEEE International Conference on Image Processing*, 1998.

[105] Y. Cui and K. Nahrstedt, "Layered Peer-to-Peer streaming," in *Proc. ACM NOSSDAV*, 2003.

[106] N. Magharei and R. Rejaie, "Adaptive Receiver-Driven Streaming from Multiple Senders," *ACM/SPIE Multimedia Systems Journal*, 2006.

[107] J. Liu, B. Li, and Q.-Q. Zhang, "Adaptive video multicast over the internet," *IEEE Transactions on Multimedia*, 2003.

[108] Y. Shen, Z. Liu, S. Panwar, K. Ross, and Y. Wang, "Streaming layered encoded video using peers," in *Proc. IEEE ICME*, 2005.

[109] D. P. T Karagiannis, P. Rodriguez, "Should internet service providers fear peer-assisted content distribution?" in *Proc. ACM/USENIX Internet Measurement Conference*, 2005.

[110] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P iptv system," *Transaction on Multimedia*, 2007.

[111] H. Xie, R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4p: Provider portal for (P2P) applications," in *Proc. ACM SIGCOMM*, 2008.

[112] V. Aggarwal, S. Bender, A. Feldmann, and A. Wichmann, "Can ISPs and P2P users cooperate for improved performance?" *ACM SIGCOMM Computer Communication Review*, 2007.

[113] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: A practical approach to reducing cross-ISP traffic in Peer-to-Peer systems," in *Proc. ACM SIGCOMM*, 2008.

[114] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in bittorrent via biased neighbor selection," in *Proc. IEEE ICDCS*, 2006.

[115] J. Li and K. Sollins, "Exploiting autonomous system information in structured Peer-to-Peer networks," in *Proc. ICCCN*, 2004.

[116] A. Nakao, L. Peterson, and A. Bavier, "A routing underlay for overlay networks," in *Proc. ACM SIGCOMM*, 2003.

[117] F. Picconi and L. Massoulie, "ISP-friend or foe? making P2P live streaming ISP-aware," in *Proc. IEEE ICDCS*, 2009.

[118] D. Tomozei and L. Massoulie, "Flow control for cost-efficient Peer-to-Peer streaming," in *Proc. IEEE INFOCOM*, 2010.

[119] N. Magharei and R. Rejaie, "Peer-to-Peer receiver-driven mesh-based streaming: Design and Evaluation," Tech. Rep. CIS-TR-06-05, 2006. [Online]. Available: http://mirage.cs.uoregon.edu/pub/tr06-05.pdf

[120] B. Alfonsi, "I want my iptv: Internet protocol television. predicted a winner," in *Proc. IEEE Distributed Systems Online*, 2005.

[121] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, "GnuStream: A P2P Media Streaming System Prototype," in *Proc. IEEE ICME*, 2003.

[122] B. Li, S. Xie, Y. Qu, G. Keung, J. Liu, C. Lin, and X. Zhang, "Inside the new coolstreaming: Principles, measurements and performance implications," in *Proc. IEEE INFOCOM*, 2008.

[123] C. Dana, D. Li, D. Harrison, and C. Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *Proc. IEEE Workshop on Multimedia Signal Processing*, 2005.

[124] R. Rejaie, M. Handley, and D. Estrin, “RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet,” in *Proc. IEEE INFOCOM*, 1999.

[125] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-based congestion control for unicaqt applications,” in *Proc. ACM SIGCOMM*, 2000.

[126] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: An Approach to Universal Topology Generation,” in *Proc. International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, 2001.

[127] T. Bonald, L. Massoulie, F. Mathieu, D. Perino, and A. Twigg, “Epidemic live streaming: Optimal performance trade-offs,” in *Proc. ACM SIGMETRICS*, 2008.

[128] T. M. Gil, F. Kaashoek, J. Li, R. Morris, and J. Stribling, “King dataset,” 2004. [Online]. Available: http://pdos.csail.mit.edu/P2Psim/kingdata/

[129] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin, “A hierarchical characterization of a live streaming media workload,” Boston University, Tech. Rep., 2002. [Online]. Available: citeseer.nj.nec.com/507862.html

[130] Y. Zhou, D. M. Chiu, and J. C. S. Lui, “A simple model for analyzing P2P streaming protocols,” in *Proc. IEEE ICNP*, 2007.

[131] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin, “A hierarchical characterization of a live streaming media workload,” in *Proc. IMW*, 2002.

[132] D. Stutzbach, R. Rejaie, and S. Sen, “Characterizing unstructured overlay topologies in modern P2P file-sharing systems,” *IEEE/ACM Transactions on Networking*, 2008.

[133] Y. Chu, A. Ganjam, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, “Early experience with an internet broadcast system based on overlay multicast,” in *Proc. USENIX Conference*, 2003.

[134] T. Ngan, D. S. Wallach, and P. Druschel, “Incentives-compatible Peer-to-Peer multicast,” in *Proc. Workshop on the Economics of Peer-to-Peer systems*, 2004.

[135] D. R. Fulkerson, “Zero-one matrices with zero trace,” *Pacific J. Math.*, 1960.

[136] A. DellEra, “Expected distinct values when selecting from a bag without replacement,” 2007.

[137] C. Huang, J. Li, and K. W. Ross, “Can internet video-on-demand be profitable?” in *Proc. ACM SIGCOMM*, 2007.

[138] C. Liang, Y. Guo, and Y. Liu, “Is random scheduling sufficient in P2P video streaming?” in *Proc. IEEE ICDCS*, 2008.

[139] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, “ Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing,” *ACM/IEEE Transactions on Networking*, 1997.

[140] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, “Randomized decentralized broadcasting algorithms,” in *Proc. IEEE INFOCOM*, 2007.

[141] C. Feng, B. Li, and B. Li, “Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits,” in *Proc. IEEE INFOCOM*, 2009.

[142] B. Q. Zhao, J. C. S. Lui, and D. M. Chiu, “Exploring the optimal chunk selection policy for data-driven P2P streaming systems,” in *Proc. Conference on Peer-to-Peer Computing*, 2009.

[143] B. Li, Y. Keung, S. Xie, F. Liu, Y. Sun, and H. Yin, “An empirical study of flash crowd dynamics in a P2P-based live video streaming system,” in *Proc. IEEE GLOBECOM*, 2008.

[144] F. Liu, B. Li, L. Zhong, B. Li, and D. Niu, “How P2P streaming systems scale over time under a flash crowd?” in *Proc. International Workshop on Peer-to-Peer Systems*, 2009.