LARGE-SCALE PEER-TO-PEER STREAMING: MODELING, MEASUREMENTS, AND OPTIMIZING SOLUTIONS

BY

Chuan Wu

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy,
Department of Electrical and Computer Engineering,
at the University of Toronto.

Copyright © 2008 by Chuan Wu.
All Rights Reserved.

Large-Scale Peer-to-Peer Streaming: Modeling, Measurements, and Optimizing Solutions

Doctor of Philosophy Thesis
Edward S. Rogers Sr. Dept. of Electrical and Computer Engineering
University of Toronto

by Chuan Wu
July 2008

Abstract

Peer-to-peer streaming has emerged as a killer application in today's Internet, delivering a large variety of live multimedia content to millions of users at any given time with low server cost. Though successfully deployed, the efficiency and optimality of the current peer-to-peer streaming protocols are still less than satisfactory. In this thesis, we investigate optimizing solutions to enhance the performance of the state-of-the-art mesh-based peer-to-peer streaming systems, utilizing both theoretical performance modeling and extensive real-world measurements. First, we model peer-to-peer streaming applications in both the single-overlay and multi-overlay scenarios, based on the solid foundation of optimization and game theories. Using these models, we design efficient and fully decentralized solutions to achieve performance optimization in peer-to-peer streaming. Then, based on a large volume of live measurements from a commercial large-scale peer-to-peer streaming application, we extensively study the real-world performance of peer-to-peer streaming over a long period of time. Highlights of our measurement study include

the topological characterization of large-scale streaming meshes, the statistical characterization of inter-peer bandwidth availability, and the investigation of server capacity utilization in peer-to-peer streaming. Utilizing in-depth insights from our measurements, we design practical algorithms that advance the performance of key protocols in peer-to-peer live streaming. We show that our optimizing solutions fulfill their design objectives in various realistic scenarios, using extensive simulations and experiments.

Acknowledgments

I would like to first express my sincere gratitude to my thesis supervisor, Professor Baochun Li, for his invaluable guidance and strong support throughout my academic life in the Ph.D. stage. I deeply appreciate his strict training and precious advice in all aspects of my academic development, which are life-long treasures for myself. I feel very fortunate to have Prof. Li as my Ph.D. supervisor, and cherish all the opportunities to explore my favorite research topics and exert my own strengths, that were only made possible with Prof. Li's unwavering support. His dedication, diligence and positiveness have always impressed me, and will continue to inspire me in my future career.

I would also like to thank all my colleagues in the iQua research group, for the valuable discussions on research, pleasant gatherings on occasions, and precious friendships that support me throughout the four years. I wish them all the best in pursuing their dreams in the future.

Finally, I feel so blessed to have a wonderful family that unconditionally supports me in my efforts towards my goals: my father and mother, my sister and brother-in-law, and their smart son, who brings so many laughters to the family. Especially, I would like to thank my beloved husband, Kien Kiong. Without his relentless support, I would not have been so focused and firm in pursuing my study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Peer-to-Peer Streaming Challenges

Based on the peer-to-peer (P2P) communication paradigm [26], live P2P multimedia streaming applications have been designed and deployed in the Internet in recent years. Not relying on any router support as required by IP-layer multicast, P2P streaming utilizes the bandwidth contribution by regular end hosts (peers) at the edge of the Internet, and greatly reduces the load on dedicated streaming servers. Therefore, P2P streaming is able to achieve much better scalability than the traditional streaming applications based on the server-client model.

Categorized on the topology and philosophy of media distribution, P2P streaming applications can be divided into three generations. The first generation features a single-tree multicast topology and push-based media dissemination along the tree [17, 26, 31, 81], which is quickly taken over by the second-generation streaming solutions, due to the vulnerability of a single multicast tree. The second-generation streaming protocols construct multiple multicast trees, and push different sub streams of each media session

down each of the trees [24, 69]. The multi-tree approaches render better failure resilience and bandwidth utilization, however, they still suffer the high cost of tree maintenance in the highly dynamic P2P networks. To fundamentally overcome the disadvantages of tree-based solutions, the state-of-the-art third-generation P2P streaming systems are built upon random mesh topologies and a distribution philosophy that features mutual *exchange* of available media blocks among peers [50, 62, 70, 87, 88].

In a third-generation mesh-based P2P streaming system, hereafter referred to as *mesh-based P2P streaming* for short, the media stream in a media session is divided into a series of consecutive blocks, and the blocks are distributed across the network by allowing neighboring peers to exchange their available blocks, that are received and cached in their local playback buffers. Each peer is connected to a number of neighbors, which are randomly selected from all the existing peers in the session, thus rendering a random mesh distribution topology. Replacing structured trees with random meshes, the mesh-based P2P streaming systems are able to achieve fundamentally better resilience to peer dynamics, more efficient use of peer bandwidths, better scalability to accommodate large flash crowds, as well as simplicity with respect to topology maintenance. Based on this design philosophy, a number of commercial P2P live streaming applications have been developed and deployed over the Internet today, which stream hundreds of media channels to millions of users at any given time. Prominent examples include CoolStreaming [88], PPLive [5], UUSee [10], TVAnts [9], PPStream [6], Joost [2], and SopCast [8].

While mesh-based streaming applications have been successfully deployed, practical experiences still show a less-than-satisfactory streaming performance at the users, with respect to long buffering delay and frequent playback interruptions, which are especially evident in case of sudden increase of the number of users in the system, *i.e.*, the flash

crowd scenarios. What is a fundamental challenge that limits the advance of P2P streaming performance? We believe the answer lies at the limited and time-varying bandwidth availability in a P2P network:

- ▷ Bandwidth is an all-time scare resource in P2P streaming. Nodes in P2P networks reside at the edge of the Internet, leading to limited per-node availability of upload capacities. Depending on the type of connection, the available per-node bandwidth also differs significantly, by at least an order of magnitude. On the other hand, to reduce buffering delay and ensure smooth streaming playback, a high streaming bandwidth, that matches the streaming bit rate of the media session, must be provided to each peer on a continuous basis. All these lead to the critical requirement to maximize the utilization of limited bandwidths in a P2P streaming network.

- ▷ Bandwidth availability is consistently changing over time. P2P networks are inherently dynamic and unreliable: Peers may join and depart at will; P2P connections can be torn down and established again between different pairs of peers all the time; the available upload capacity at each peer and that along a P2P link are varying over time as well. All such volatility constitutes a great obstacle to achieving stable streaming bit rates in P2P streaming sessions.

While mesh-based P2P streaming outperforms tree-based streaming solutions in addressing the above challenge, it is interesting to observe that the existing mesh-based solutions typically employ heuristic designs for their key protocols, including the random peer selection strategies and ad-hoc server capacity deployment methods. All these facts motivate us to investigate: Can we do better in the design of mesh-based P2P streaming, such that an optimized streaming performance can be provided to all the peers at all times?

1.2 Thesis Scope

This thesis focuses on fundamental improvement of key protocols in mesh-based P2P live streaming from multiple aspects, towards the ultimate goal of enhancing the performance of P2P streaming. Our study can be categorized into two broad categories: From the more theoretical perspective, we carefully model mesh-based P2P streaming using optimization and game theoretical models, and design efficient and fully decentralized solutions to achieve performance optimization in various practical scenarios; from the more practical perspective, we extensively measure and analyze a real-world large-scale P2P streaming application using a large volume of traces collected over a one-year period of time, and design practical enhancement solutions based on in-depth insights from the measurements.

In the first category of our study, we focus on the optimization of bandwidth utilization in a P2P streaming network. Aiming to minimize end-to-end latencies in a single streaming overlay, we model the minimum-delay peer selection and bandwidth allocation problem into a global linear optimization model. We then decompose the problem into distributed subproblems using effective algorithmic techniques, and propose efficient algorithms to solve the subproblems in a fully decentralized fashion. In a more practical scenario without any assumption of *a prior* knowledge of link or node bandwidths, we further investigate the streaming rate satisfaction problem, by formulating it into a linear feasibility problem and designing a practical distributed solution algorithm. In our distributed algorithm designs, each peer carries out local optimization steps for bandwidth allocation on its neighboring links, and the entire system can be proven to converge to the global optimum or streaming rate feasibility on an ongoing basis.

Beyond the single overlay optimization, we further investigate the challenge when

coexisting streaming overlays are considered, corresponding to different channels of programming, which are the norm in a typical P2P streaming application. In such a practical system, a typical scenario persists, that multiple peers from different overlays compete for limited upload bandwidth at their common streaming servers or upstream peers in the network. To resolve such a bandwidth conflict, we model the bandwidth competition in a game theoretic setting, with a distributed collection of dynamic auction games, where downstream peers from different overlays bid for upload bandwidth at the same upstream peer. We then design cleanly decentralized strategies to carry out the bandwidth auctions, which converge to global optimal streaming topologies for all the overlays in realistic asynchronous environments.

In the second category of our study, we extensively measure and analyze a large-scale P2P streaming application, in collaboration with UUSee Inc. [10], one of the leading P2P live streaming solution providers based in Beijing, China. With collaborative efforts, we instrument the application to collect large volumes of traces, which amount to almost a terabyte over a span of one year, involving millions of users, with a snapshot of the entire system every five minutes. Using this large volume of live traces, our in-depth study of this practical system features the following highlights.

First, as mesh streaming topologies play an important role towards the commercial success of P2P streaming, it is critical to acquire a thorough and in-depth understanding of the topological characteristics of these large-scale meshes, their time-varying behavior, and how they react to extreme scenarios such as flash crowds. Based on 120 GB of traces collected over a time span of two months, we analyze the graph theoretic properties of UUSee streaming topologies, with respect to their degrees, clustering coefficients, reciprocity, and likelihood. We have drawn conclusions with respect to the scalability,

clustering, and reciprocal effects of the state-of-the-art P2P streaming topologies, based on our observations.

Second, in a P2P streaming application, it is important for peers to select an appropriate number of neighboring peers with high-bandwidth active connections. It would then be helpful to obtain some kind of approximate knowledge of available TCP bandwidth between any two peers without active probing. To achieve this objective, we have conducted an exhaustive investigation with respect to statistical properties of TCP throughput values on streaming flows among peers, using 230 GB of UUSee traces over a four-month period of time. We make quite a number of interesting discoveries with respect to the key factors that decide inter-peer bandwidth availability. Based on our discoveries, we design a throughput expectation index that makes it possible to select high-bandwidth peers without probing.

Finally, we have monitored the streaming quality in different UUSee streaming channels over a seven-month span, and investigated the server capacity consumption on all the UUSee streaming servers, based on 400 GB of traces collected during this period. We have made an intriguing discovery that the deployed server capacities are not able to keep up with the rapidly increasing demand in the streaming channels in such a typical system. This motivates us to design an online server capacity provisioning algorithm, which dynamically allocates the pool of available server capacity to each of the concurrent channels to address their predicted demand, and also guides the deployment of an appropriate amount of total server capacity in the system. As another important objective, we also consider the minimization of inter-ISP (Internet Service Provider) traffic in our design, by carrying out the algorithm with full ISP-awareness to maximally constrain P2P traffic within ISP boundaries.

1.3 Main Contributions

The main contributions of the thesis work are summarized as follows.

- We design a decentralized optimal peer selection and bandwidth allocation algorithm, that achieves minimized end-to-end latencies at all the participating peers in a streaming session. Our design is firmly based on the Lagrangian dual theory in optimization.

- We design a distributed protocol to effectively carry out streaming bandwidth allocation in the most practical scenario without any *a prior* knowledge of node or link bandwidth availability. We rigidly prove that the algorithm is able to achieve the required streaming rate at all peers in a streaming session in both synchronous and asynchronous dynamic environments.

- Previous studies in P2P streaming have mostly focused on a single overlay, neglecting the conflicting nature among coexisting overlays with respect to available network bandwidth. We present the first study in the literature that focuses on such conflicts, by modeling them as a distributed collection of bandwidth auction games, and by designing efficient tactical strategies to carry out the bandwidth auctions. Our design is further different from most existing work on game theory, in that we utilize the auction game model to facilitate the design of a simple, practical and fully decentralized protocol that achieves global properties. Our extensive theoretical analysis shows that the decentralized auction strategies not only converge to a Nash equilibrium in realistic asynchronous environments, but also lead to an optimal topology for each of the coexisting streaming overlays.

- The scale of our measurement work with respect to UUSee is unprecedented in P2P streaming research, with a terabyte worth of live traces collected over a span of one year and a snapshot of the entire system every five minutes.

- We are the first to thoroughly investigate the topological properties of large-scale P2P streaming meshes, and their evolutionary characteristics. The original insights we have brought forward in this study include: (1) Topologies of P2P streaming meshes do not possess similar properties as those obtained from early Internet or AS-level topological studies, such as power-law degree distributions; (2) The streaming topologies naturally evolve into clusters inside each ISP, but not within geographically adjacent areas; (3) Peers are reciprocal to each other to a great extent, which contributes to the stable performance of mesh-based streaming; (4) There do not exist super-node hubs in the streaming topologies.

- Our in-depth statistical characterization of P2P streaming flows has led to a number of interesting discoveries: (1) The ISPs that peers belong to are highly relevant in determining inter-peer bandwidth, even more important than their geographic locations; (2) Excellent linear correlations exist between the availability of peer last-mile bandwidth and inter-peer bandwidth within the same ISP and between a subset of ISPs, with different linear regression coefficients for different pairs of ISPs; (3) Inter-peer bandwidth exhibits an excellent daily evolutionary pattern within or across most ISPs.

- We design a throughput expectation index, which facilitates a new way of selecting high-bandwidth serving peers without any active probing for available bandwidth. Instead, this index is computed based on the ISPs that peers belong to, a table

of linear regression coefficients for different pairs of ISPs, the availability of peer last-mile bandwidth, and the time of the day.

- Controversial to most common beliefs, our measurement study has revealed the indispensable role of dedicated streaming servers, in providing stable upload capacities to accommodate peer instability in real-world P2P streaming applications. In particular, we discover that the ad-hoc way of server capacity provisioning on the streaming servers is not able to effectively accommodate the dynamic demand in the streaming channels, not to mention that with an increasing number of channels deployed.

- We design an online server capacity provisioning algorithm, that dynamically allocates the pool of available server capacity to each of the concurrent channels, and also possesses full ISP-awareness to maximally constrain P2P traffic within ISP boundaries. The algorithm works in a proactive fashion, that optimally allocates the server capacity among concurrent channels based on their predicted popularity, forecasted server capacity demand, and the priority of each channel. It can also be utilized to guide the provisioning of total server capacity in each ISP over time. We validate the effectiveness of the algorithm based on a realistic replay of the trace scenarios in a P2P streaming system that emulates UUSee protocols.

1.4 Thesis Organization

The remainder of the thesis is organized as follows. We present background materials and related research in Chapter 2. In Chapter 3, we discuss our minimum-delay bandwidth allocation model, and the design of a distributed optimization algorithm. In Chapter 4,

we model the streaming rate feasibility problem, and propose an efficient algorithm to achieve it without any *a prior* knowledge of link or node bandwidths. We then characterize the bandwidth conflict among coexisting streaming overlays in Chapter 5, and present our design of effective conflict-resolving strategies based on an auction game model. We start to present our measurement-based studies of a practical large-scale P2P streaming application from Chapter 6, in which we describe our measurement methodology and basic global characteristics summarized from the traces. We then present our topological characterization of large-scale streaming meshes in Chapter 7, statistically study inter-peer bandwidths and design a new way of selecting high-bandwidth peers in Chapter 8, and investigate server capacity consumption that motivates the design of an online server capacity provisioning algorithm in Chapter 9. Finally, we present concluding remarks and future directions in Chapter 10.

Chapter 2

Background

2.1 P2P Streaming Systems

Since the seminal work on end system multicast in 2000 [26], P2P streaming has emerged as one of the most successful applications of the overlay multicast paradigm. The development of P2P streaming systems can be categories into three generations.

2.1.1 Single-tree P2P streaming

The first-generation P2P streaming solutions feature the streaming topology of a single multicast tree, with representative systems as follows.

Narada [26] is among the first multicast systems that advocate multicast among end hosts at the application layer, rather than among routers in the network layer. In Narada, a multicast tree is established by running a distant vector routing algorithm to decide the shortest paths between the streaming source and the receivers. Its tree construction further provides the flexibility of constructing based on different performance metrics,

such as low delay, low data loss rate, or high bandwidth.

ZIGZAG [81] organizes network nodes into an application-layer multicast tree, with a height logarithmic with the number of receivers and a node degree bounded by a constant. This tree structure reduces the number of processing hops on the delivery path to a receiver, avoids network bottlenecks, and keeps the end-to-end delay from the source to a receiver small.

Other single-tree P2P streaming systems include SpreadIt [31] and NICE [17]. These single-tree solutions suffer sub-optimal performance of throughput, as the media stream is pushed down the multicast tree using a same streaming rate. In addition, as each interior node uploads to multiple downstream nodes, the challenge further surfaces when they depart or fail, which interrupts the streaming session and requires expensive repair processes.

2.1.2 Multi-tree P2P streaming

To overcome those disadvantages, the second-generation P2P streaming protocols construct multiple multicast trees, each rooted at the streaming server and spanning all the nodes.

CoopNet [69] employs multiple description coding (MDC) to support its multi-tree streaming. With MDC coding, the streaming media content is divided into multiple substreams, referred to as descriptions, and each description is delivered to the receivers via a different multicast tree. Every subset of the descriptions is decodable. In this way, each node decides how many multicast trees to join, i.e., how many descriptions to receive, based on its download bandwidth.

SplitStream [24] is another P2P system focusing on high-bandwidth multicast based

on multiple trees. To achieve load balancing on the small number of interior nodes, it constructs a forest of interior-node-disjoint multicast trees that distribute the forwarding load among all the participating peers. The data stream is divided into multiple blocks, and each block is distributed along a different tree.

As compared to single-tree solutions, these systems can better accommodate peers with heterogeneous bandwidths by having each peer join different numbers of trees. It is also more robust to peer departures and failures, as an affected peer may still be able to continuously receive the media at a degraded quality from other trees, while waiting for the affected tree to be repaired. These advantages come with a cost, however, as all the trees need to be maintained in highly dynamic P2P networks.

2.1.3 Mesh-based P2P streaming

To provide fundamentally better flexibility, robustness, and bandwidth efficiency, the third-generation mesh-based P2P streaming systems have been designed and deployed in recent years.

Bullet. In the Bullet [50] system, nodes self-organize into a high-bandwidth overlay mesh, and data blocks are distributed in a disjoint manner to strategic points in the network. Individual receivers locate the data blocks using RanSub [49], a distributed approach that uniformly spreads the availability information of data blocks across all the overlay participants, and retrieve the data from multiple points in parallel.

CoolStreaming [88] constructs a data-driven mesh overlay network for live media streaming. It also represents the first mesh-based P2P streaming system deployed over the Internet in large scale. In CoolStreaming, the media stream in each media session is divided into consecutive blocks, and is distributed across the network by allowing peers

to exchange their available blocks, that are received and cached in their local playback buffers. The playback buffer at each peer represents a sliding window of the media channel, containing blocks to be played in the immediate future. Each peer has a list of partners for media block exchange, whose information is acquired using a gossip protocol. A peer periodically exchanges the media block availability information in its playback buffer, *i.e.*, its *buffer availability bitmap* or *buffer map* in short, with its partners, and retrieves unavailable data from one or more partners.

Chainsaw [70] is another multicast system that is based on a randomly constructed P2P mesh graph with a fixed minimum node degree. In Chainsaw, when a peer receives a new packet, it sends a NOTIFY message to its neighbors. Every peer maintains a window of interest, and explicitly requests a packet from a neighbor in order to receive it.

GridMedia [87] employs a hybrid push-pull approach on its random mesh topology for low-delay P2P live streaming. After a peer joins the GridMedia system, in the first time interval, it requests ("pulls") packets from its neighbors. In what follows, at the end of each time interval, it subscribes the packets to retrieve at each neighbor, and the neighbors will relay ("push") the packets to the peer as soon as they receive the packets. They argue that while the pull mode enables dynamic connectivity in the highly dynamic P2P environment, the push mode can effectively reduce the accumulated latency observed at the receiver nodes.

PRIME [62] is a recent mesh-based P2P live streaming protocol that considers the bandwidth bottleneck and content bottleneck in the swarming protocol design. In PRIME, incoming and outgoing degrees of individual peers are decided to minimize the

probability of bandwidth bottleneck in the content delivery. To minimize content bottleneck, PRIME summarizes the content delivery pattern into a diffusion phase and a swarming phase, and designs a packet scheduling algorithm to accommodate this pattern.

Streaming over random mesh topologies, these systems can achieve more efficient utilization of the limited bandwidth in the P2P network, attributed to the simultaneous parallel downloads. The mutual block exchange and loosely-coupled P2P connectivity provide simplicity and flexibility, that adapt better to the volatile peer dynamics in a P2P network. Based on this design philosophy, a number of real-world P2P streaming applications have been implemented and deployed in the Internet, including PPLive [5], UUSee [10], TVAnts [9], PPStream [6], Joost [2], SopCast [8], Zattoo [12], VVSky [11], QQLive [7].

2.2 Optimization in P2P Resource Allocation

2.2.1 Mathematical programming

Many network resource utilization problems are optimization problems in nature. Optimization is the approximate synonym for *mathematical programming* (MP) in operations research, where mathematical programming is the study of optimization problems in which one seeks to minimize or maximize a real function of real variables, subject to constraints on these variables. The general form of a mathematical programming problem is as follows, which is also referred to as a *mathematical program*:

$$\min f(x) \tag{2.1}$$

subject to

$$g_i(x) \leq b_i, i = 1, \ldots, m. \tag{2.2}$$

Here, $x \in \mathbb{R}^n$ is the *optimization variable* vector. The function $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, and the functions $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$, are the *constraint functions*.

If the objective function f and constraint functions $g_i, i = 1, \ldots, m$, are convex functions [23], the mathematical optimization problem is a convex optimization problem, and the mathematical program in (2.1) is referred to as a *convex program* [23]. Convex programming is an important branch of mathematical programming, as it can be used to express many practical optimization problems and is algorithmically tractable. As a linear function is convex, *linear programming*, in which the objective function and constraint functions are linear, is an important category of convex programming, which can model a broad range of optimization problems in practice.

Convex programming is a well-studied category of optimization, with a number of algorithms invented to solve a convex program. These algorithms include the simplex algorithm [30], ellipsoid method [47, 48, 74, 86], interior point methods [33, 46], and subgradient algorithms [18, 75].

2.2.2 Lagrangian relaxation and subgradient algorithm

Most of the convex programming algorithms are inherently centralized algorithms, which require all the computations to be carried out in one computer with all the required information. There is one category of optimization techniques, the Lagrangian relaxation and subgradient algorithm, which explores the decomposable structure of an optimization problem, and renders distributed solution algorithms.

Lagrangian relaxation is an important mathematical programming technique [18, 23], that relaxes a set of constraints and generates the *Lagrange dual* of the original optimization problem, *i.e.*, the *primal* problem. To relax a set of constraints $g_k,\ \forall\ k \in \mathcal{K}, \mathcal{K} \subset \{1, 2, \ldots, m\}$, we associate a Lagrange multiplier, μ_k, with each of the constraints, and incorporate them into the objective function, which then becomes $f(x) + \sum_{k\in\mathcal{K}} \mu_k(g_k - b_k)$. We can derive the Lagrange dual as

$$\max_{\mu} L(\mu) \tag{2.3}$$

subject to

$$\mu_k \geq 0,\ \forall\ k \in \mathcal{K},$$

where

$$L(\mu) = \min_{P} f(x) + \sum_{k\in\mathcal{K}} \mu_k(g_k - b_k), \tag{2.4}$$

and P is a polytope defined by the rest of the constraints:

$$g_i(x) \leq b_i,\ \forall\ i \in \{1, \ldots, m\} - \mathcal{K}.$$

The Lagrange Duality theorem states that, the optimal value of (2.4) is always an lower-bound of the optimal value of the original problem in (2.1). When the primal problem is convex, if we take the maximum of the new optimal values of (2.4) over all non-negative values of $\mu_k,\ \forall\ k \in \mathcal{K}$, the derived optimal objective function value of the Lagrange dual problem in (2.3) is exactly the same as the optimal value of the original problem.

The optimal solution to the Lagrange dual can be derived with the subgradient algorithm [18, 75]. The subgradient algorithm derives the optimal values of the Lagrange multipliers in an iterative fashion: In each step, given the current non-negative values of the multipliers, it computes the optimal solution to the problem in (2.4), and then updates the multiplier values along the subgradient direction according to a sequence of prescribed step lengths. The subgradient algorithm converges to optimal values of μ, *i.e.*, μ^*. However, for linear programs, the optimal values of x derived using the optimal μ^* from (2.4) are not necessarily the optimal variable values for the primal problem in (2.1) [73]. In this case, a primal recovery algorithm can be applied to derive the primal optimal variables [73].

By choosing an appropriate set of constraints to relax, the derived Lagrange dual problem may represent a decomposable structure, which can be exploited to decompose it into subproblems that can be efficiently solved in a distributed manner. In this way, Lagrangian relaxation constitutes a powerful tool to design efficient and distributed solution algorithms to solve the original optimization problems.

2.2.3 Application

Optimization techniques have been utilized to model a variety of problems in P2P content distribution, including the optimal allocation of flow rates to achieve maximum throughput, minimum cost, or fairness in overlay multicast, and the optimal peer selection and topology construction in P2P applications.

For overlay multicast over a single-tree topology, Cui *et al.* [28] formulate the optimal flow rate computation problem into a non-linear program, with an aggregate utility objective function and flow rates on the overlay links as the optimization variables. Both

network capacity constraints and data constraints are defined to guarantee a valid multicast flow. They apply Lagrangian relaxation technique to derive a distribution solution algorithm, in which overlay nodes coordinate to accomplish the link rate computation tasks.

In the more complicated scenario of multicast in a general data network, Garg *et al.* [34] study the maximum throughput problem by constructing optimal multicast trees. In their linear program, the variables indicate flow rates along different multicast trees, and the objective is to maximize the overall throughput along all possible multicast trees in the network, subject to link capacity constraints. As this formulation is equivalent to a NP-hard fractional packing problem, they design an approximation algorithm to solve the linear program.

In order to solve this maximum throughput problem in polynomial time, Li *et al.* [58] introduce conceptual flows to model the data constraint in a valid multicast flow, rather than resorting to tree packing. In their formulation, the multicast flow from the server to all the receivers is viewed as composing of multiple conceptual unicast flows from the server to each of the receivers. Each of the conceptual unicast flows follows flow conservation, and the multicast rate on each overlay link is the maximum of the rates of all conceptual flows going along this link. Based on this notion, Li *et al.* formulate a linear program to achieve maximum multicast throughput, by adjusting rates of those conceptual flows. They apply Lagrangian relaxation to decompose the problem into k min-cut problems, based on which a polynomial-time distributed algorithm is achieved.

Another practical objective in computing optimal flow rate allocation is to achieve the best fairness among all the receivers. Cui *et al.* [29] formulate a variation of linear programs to model the max-min fairness within one overlay multicast tree, with the

objective function in the form of a max-min rate vector. A distributed algorithm is designed to derive the solution in the case that the multicast tree topology is given.

For overlay multicast applications with an end-to-end rate requirement, such as media streaming applications, the optimization objective generally represents the minimization of the cost incurred when transmitting the data at the required rate, rather than the maximization of throughput. In the optimization model of Lun *et al.* [60], the objective function is a cost function with variables denoting multicast rates on each overlay link. With linear cost functions, the problem is a linear program with the embedded structure of minimum-cost flow problems. Lun *et al.* exploit this embedded structure by applying Lagrangian relaxation to constraints not belonging to the minimum-cost flow formulation.

In P2P file sharing applications, optimization techniques have been utilized to decide the best set of supplying peers for each receiving peer, and the corresponding downloading rates [13][44]. In Adler *et al.*'s model [13], the total downloading delay is formulated as the optimization objective, while a total downloading budget and the available downloading bandwidth from each peer constitute constraints. A similar optimization model for peer selection is proposed by Janakiraman *et al.* [44]. This model has an objective function in the non-standard minimax form, but can be transformed into a linear or convex program, which can then be solved numerically by standard linear/convex programming algorithms.

In P2P streaming, the optimization models for optimal peer selection [13][40] address the specific requirements of media streaming applications, such as the Quality of Service (QoS) requirements in terms of delay, jitter, etc. For example, in Adler *et al.*'s non-linear optimization model [13], the optimization objective is the minimization of total streaming cost, while a continuous playback is guaranteed with the constraints.

2.3 Game Theory in P2P Applications

2.3.1 Game theory preliminaries

In game theory, a *noncooperative strategic game* is a model of interactive decision-making, in which each decision-maker chooses his plan of action once and for all, and these choices are made simultaneously [67]. A strategic game G consists of the following key elements:

- a finite set N (the set of *players*)
- for each player $i \in N$, a nonempty set A_i (the set of *actions* available to player i)
- for each player $i \in N$, a preference relation $\succsim_i$ on $A = \times_{j \in N} A_j$ (the *preference relation* of player i)

In a strategic game where each player makes their own decision based on their preference relations, the most commonly used solution concept is the *Nash equilibrium*. Nash equilibrium captures a steady state of the play of a strategic game, in which each player holds the correct expectation about the other players' behavior and acts rationally. More specifically, a Nash equilibrium of a strategic game $< N, (A_i), (\succsim_i) >$ is a profile $a^* \in A$ of actions, with the property that for every player $i \in N$, we have $(a^*_{-i}, a^*_i) \succsim_i (a^*_{-i}, a_i)$ for all $a_i \in A_i$. Therefore, at a Nash equilibrium a^*, no player i has an action yielding an outcome that he prefers to that generated when he chooses a^*_i, given that every other player j chooses his equilibrium action a^*_j. In another word, no player can profitably deviate, given the actions of the other players.

When the players repeatedly engage in a strategic game G for numerous times, the situation is captured by the model of a *dynamic*, or *repeated* game [67]. In a dynamic

game, players have some information about the strategies chosen by others previously, and thus may contingent their play on the past moves.

In a strategic game where noncooperative players share a common resource, a *price of anarchy* is defined to measure the effectiveness of the outcome of the system [51]:

$$\text{Price of Anarchy} = \frac{\text{Cost of Worst Nash Equilibrium}}{\text{Socially-Optimal Cost}}.$$

The price of anarchy measures how much performance is lost due to the lack of coordination among selfish players in a game model, *i.e.*, how far the outcome of the game is from the social optimum, defined in terms of a specific performance metric, such as minimal total delay in an Internet routing system. A Nash equilibrium is *efficient* if it coincides with the solution to a global optimization problem reflecting the social optimum.

2.3.2 Application

In P2P applications, noncooperative strategic games have been exploited to characterize peer selfishness or competition, and to provide incentives that motivate peers to contribute their upload capacities or efficiently share common bandwidth resources [37, 53, 61, 80].

Fabrikant *et al.* [32] introduce a network creation game, in which selfish nodes choose a subset of other nodes and pay for the links that they establish, with the goal of minimizing its connecting cost and distance to every other node in the network. The outcome of the game is the network topology among the nodes. They show that the price of anarchy in this network creation game, i.e., the relative cost of the lack of coordination among selfish nodes, may be modest.

Chun *et al.* [27] further the above study by modeling the construction of overlay routing networks as selfish nodes playing competitive network construction games. They find that by varying the link cost function, the game produces widely different Nash equilibria, *i.e.*, different topologies that achieve diverse goals, such as low stretch, balanced degree distribution, or high failure or attack resilience.

In P2P file sharing networks, Golle *et al.* [37] examine the free-riding problem, *i.e.*, peers obtain services without themselves contributing any resources, by modeling the file sharing scenario during one time period using game theory. In their model, selfish peers select what proportion of files to share and determine how much to download from the network in each period, in order to maximize their own quasi-linear utility functions. They design several payment mechanisms to encourage file sharing, and analyze equilibria of user strategies under these payment mechanisms. Lai *et al.* [53] use the Evolutionary Prisoners Dilemma model to capture the tension between individual and social utility in a selfish P2P file sharing network. Each peer plays the roles as both a client and a server in each round of the dynamic game, to decide between the actions of requesting a file or not, and between allowing download and ignoring request, respectively.

To provide service differentiation among peers based on their contribution levels, Ma *et al.* [61] model the bandwidth request and distribution process as a competition game among the competing nodes in a P2P file sharing network. The upload capacity at each uploading peer is distributed based on the amount of service each of its downstream nodes has provided to the community. They show that this game has a Nash equilibrium, is collusion-proof, and maximizes the social welfare.

To encourage contribution in P2P streaming applications, Tan *et al.* [80] propose a payment-based incentive mechanism, in which peers earn points by forwarding data to

others. The streaming process is divided into fixed-length periods. During each period, peers compete for good suppliers for the next period using their points, based on strategies that maximize its own expected media quality.

A pricing mechanism is designed by Qiu *et al.* [71] to stimulate cooperation in P2P-like ad-hoc networks. In their model, network users can charge other users a price for relaying their data packets, and the prices and relay rates are set tactically to maximize their own net benefit. An iterative price and rate adaption algorithm is designed, that converges to a socially optimal bandwidth allocation in the network.

2.4 P2P Streaming Measurements

With the successful deployment of mesh-based P2P streaming applications, there have recently emerged a number of measurement studies, in which researchers collect live traces from real-world systems and investigate their performance. In the majority of existing measurement studies, traces are collected using either the *crawling* method or *passive sniffing* technique. In the former category, a *crawler* program or script is designed and deployed, that collects peer information in a P2P application by sending protocol messages to the known peers and collecting corresponding responses to learn more about other peers. In the second category, a number of dedicated computers are deployed, that actually join a P2P application as peers by running the P2P client software. They then *sniff* packet traffic from and to themselves, and collect related performance measurements.

2.4.1 PPLive

Hei *et al.* [41, 42] have carried out a measurement study of *PPLive*, one of the most popular P2P live streaming applications in the Internet. Their trace collection incorporates both active crawling and passive sniffing.

Using a PPLive crawler, they collect global information of the P2P applications during a few time periods in 2006 and 2007. The global characteristics derived include the number of users in the application and their variation over time, the geographic distribution of the users, the arrival and departure rates of users in two popular channels.

Using passive sniffing, they monitor packet traces on two dedicated nodes deployed on high-speed campus access networks and two nodes on broadband residential access networks. The local media traffic characteristics they have investigated in this way include start-up delays, playback lags, fractions of control and video data each peer receives, peer partnership, and the aggregate upload/download rates at the four peers.

The following insights are derived from this measurement study: (1) P2P streaming users have the similar viewing behaviors as regular TV users; (2) A peer exchanges video data dynamically with a large number of other peers; (3) A small set of super-peers act as video proxy and contribute significantly to video data uploading; (4) Users in the system can suffer long start-up delays and playback lags, ranging from several seconds to a couple of minutes.

In a more recent work [43], Hei *et al.* have further exploited the buffer maps harvested from the peers in PPLive to monitor the network-wide streaming quality. The buffer maps were collected over short periods (2-4 hour intervals) in 2007, using both a dedicated buffer-map crawler and 3 passive sniffing nodes. As a buffer map reflects information about the data blocks each peer makes available for sharing, they show that

information provided by the advertised buffer map of a peer correlates to that peer's viewing continuity and start-up latency. They also present results inferred from a large number of buffer maps, with respect to network-wide playback continuity, start-up latency, playback lags, and block propagation.

Also focusing on PPLive [83], Vu *et al.* investigate its two graphical properties, node outdegree and graph clustering coefficient, based on traces collected from 300 crawled peers in one streaming channel over a 24-hour period in 2006. They conclude from their study that small overlays in PPLive are similar to random graphs in structure, and the average outdegree of a peer in an overlay is independent of the population in the streaming channel.

2.4.2 CoolStreaming

Li *et al.* [54, 55] measure the CoolStreaming system, based on traces collected using an internal logging system. In their trace collection, an ActiveX component in JavaScript code is deployed at the peers in the system, which collects peer activities and status information and reports them to a log server. The peer activities, including join and departure events, are collected when they occur; the status information is collected every 5 minutes, which may include the following: the percentage of missing video data at playback deadline, the volume of video content downloaded and uploaded, and the partner information.

Using traces collected during live broadcast events in one sport channel on September 27th, 2006, they have investigated the type and distribution of users in CoolStreaming, their join/departure rates, the distribution of user session durations, the distribution of

peer uploading contributions, and the start-up delay and playback continuity in the system. From this study, they show that (1) the churn is the most critical factor that affects the overall performance of the system, (2) there is a highly unbalanced distribution in terms of uploading contribution among the peers, and (3) the excessive start-up time and high failure rates during flash crowds remain critical problems in such a P2P streaming system.

2.4.3 Others

Ali *et al.* [15] have studied PPLive and SopCast with information collected in 2-3 hour durations in October and December 2005, by passive sniffing on a number of nodes deployed in different ISPs with different types of last-mile connections. They mainly analyze the resource usage in the two applications, with respect to the total sending and receiving throughput at each peer, the number of children supported by each peer, and the locality and stability of data distribution. Based on their study, they identify a number of shortcomings and challenges these applications are likely to face: (1) Random peer selection may result in inefficient distribution. For example, a North American node may download all the way from Asia parents, but will upload back to peers in Asia. (2) The systems depend heavily on the availability of high-capacity nodes, which are richly connected to supply other peers. (3) Most communications in the systems are based on TCP, even for small amounts of data, which may not be ideal for such real-time applications due to the delay property of TCP connections. (4) NAT traversal is not efficiently handled in the two systems, such that peers behind NATs are unable to upload to other peers.

Silverston *et al.* [77, 78] compare the traffic pattern among four popular P2P streaming

applications, PPLive, PPStream, SOPCast, and TVAnts, using measurements collected by passive sniffing during the broadcast of the 2006 FIFA world cup. Four personal computers were deployed in a campus Ethernet network and a residential ADSL network, to run these applications and to collect traffic measurements when watching two live FIFA games on June 30, 2006. From each of the observing nodes running different applications, they compare their total upload/download traffic, the upload traffic from top serving peers, the types of video and control traffic involved (UDP or TCP), the number of serving peers, and the peer life time distribution. They observe diversity among all the applications, with respect to different traffic patterns, different video download policies, and various neighborhood sizes. Nevertheless, they conclude that the peer life time in all four applications follows Weibull distributions, though with different average lengths.

Chapter 3

Optimal Peer Selection for Minimum-Delay P2P Streaming

The limited bandwidth availability in P2P networks pose a significant technical challenge in P2P live media streaming, which has two key requirements: (1) A typical streaming bit rate, generated with a current-generation codec such as H.264/AVC, H.263 or MPEG-4, must be accommodated for all the peers in each streaming channel; (2) The end-to-end latency at each peer has to be minimized, in order to guarantee high liveness of the media content. To address these requirements in a mesh-based P2P streaming system, a critical question arises: What is the best way for the peers to select their upstream supplying peers and allocate the streaming rates from the selected peers, such that a specified aggregate streaming bit rate is satisfied and the end-to-end latencies are minimized at all the receivers? It is a nontrivial problem to obtain a feasible streaming rate allocation strategy which guarantees all receiving peers can acquire the streaming bit rate of the media channel, not to mention that which minimizes end-to-end latencies.

To decide streaming rates among peers, most existing P2P streaming protocols do

not employ explicit bandwidth allocation, but resort to TCP, TFRC [38] or UDP with heuristic traffic shapers [50, 87, 88]. For those using TCP or TFRC [50, 88], transmission rates are adjusted on a per-link basis, and there is no guarantee that the required aggregate streaming rate can be provided to each receiving peer. Delivering media packets over UDP, GridMedia [87] applies traffic shapers at each sending peer towards each of its downstream peers to guarantee smooth delivery, but it is not discussed how to carefully allocate the upload capacity to satisfy the rate requirement of the downstream peers.

In this chapter, we aim to address the requirements of P2P live streaming using an optimization approach. We design an optimal peer selection and streaming rate allocation algorithm, that explicitly takes the streaming bandwidth demand into consideration in the rate adjustment at each peer across all its downloading links, while guaranteeing minimized end-to-end latencies at the receivers.

To design this optimization algorithm, we first formulate the optimal peer selection problem as a linear optimization problem, which guarantees bandwidth availability and minimizes streaming latencies. We then design an efficient and decentralized algorithm to solve the problem, based on the Lagrangian relaxation technique and the subgradient algorithm. The derived optimal peer selection algorithm computes the optimal streaming rates on the P2P links in a fully decentralized and iterative fashion, and is also reactive to network dynamics, including peer joins, departures and failures.

3.1 Problem Formulation

We consider a P2P streaming session, corresponding to the broadcast of one streaming channel, with one streaming *server* and multiple participating *receivers* (Fig. 3.1). There exists a standalone neighbor list maintenance mechanism in the network, consisting of one

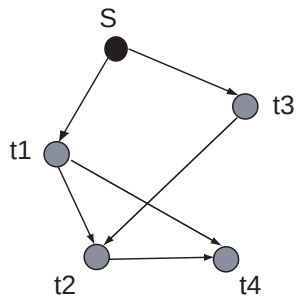
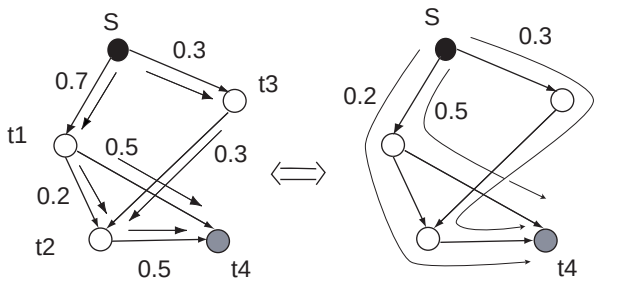


Figure 3.1: An example of the P2P streaming network model: S - the streaming server, $t1, t2, t3, t4$ - the receivers.

or multiple bootstrapping servers. When a new peer joins the session, it is bootstrapped by one of the bootstrapping servers with a list of known peers in the session, who may serve as the initial set of *upstream* peers. This constructs the initial *mesh* overlay topology for the streaming session. Such a mesh topology can be modeled as a directed graph $G = (N, A)$, where N is the set of vertices (peers) and A is the set of directed arcs (directed overlay links). Let S be the streaming server, and let T be the set of receivers in the streaming session. We have $N = \{S\} \cup T$.

With this network model, our objective is to design an optimal peer selection and streaming rate allocation strategy, which decides the best subset of upstream peers (from all the known upstream peers) to retrieve from at each receiver and the corresponding optimal streaming rates from each of them; such a strategy constructs an optimal streaming topology, on top of which the end-to-end latencies at all receivers are minimized. We formulate a linear programming model to address this optimal peer selection problem. We model the objective function to reflect the minimization of streaming latencies at the receivers, and the constraints to reflect the capacity limitations and flow constraints in the P2P session. The key challenges of the modeling lie at the formulation of end-to-end latencies in a multicast network, and the flow constraints to model a valid multicast



(a) a unicast flow (b) three fractional flows

Figure 3.2: An example of a unicast flow from S to t_4 and its decomposition into three fractional flows.

flow. In what follows, we motivate our linear program (LP) formulation of multicast P2P streaming by first analyzing a unicast streaming session from the streaming server to one receiver.

3.1.1 LP for unicast streaming

A unicast flow from the streaming server to a receiver is a standard network flow observing the property of flow conservation at each intermediate node. Let r be the streaming rate of this unicast flow, c_{ij} be the link delay and f_{ij} be the transmission rate on overlay link (i, j). Fig. 3.2(a) depicts an example of a unit unicast flow from S to t_4, with $r = 1$, $c_{ij} = 1$, $\forall\ (i, j) \in A$, and the rates f_{ij} labeled on the arcs. Such a unicast flow can be viewed as multiple fractional flows, each going along a different overlay path. Different paths may share some same overlay links, and the transmission rate on each shared link is the sum of rates of all fractional flows that go through the link. Fig. 3.2(b) illustrates the decomposition of the unit unicast flow into three fractional flows, with rates 0.2, 0.3 and 0.5, respectively.

We calculate the average end-to-end link latency of a unicast flow as the weighted

average of the end-to-end latencies of all its fractional flows, with the weight being the ratio of the fractional flow rate to the aggregate unicast flow rate. In Fig. 3.2, the end-to-end link delays of the three paths are 3, 3 and 2 respectively, and thus the average end-to-end latency is $0.2 \times 3 + 0.3 \times 3 + 0.5 \times 2$. We further notice that

$$\begin{aligned} & 0.2 \times (1+1+1) + 0.3 \times (1+1+1) + 0.5 \times (1+1) \\ = \; & 1 \times (0.2+0.5) + 1 \times 0.3 + 1 \times 0.2 + 1 \times 0.5 + 1 \times 0.3 + 1 \times (0.2+0.3) \\ = \; & 1 \times 0.7 + 1 \times 0.3 + 1 \times 0.2 + 1 \times 0.5 + 1 \times 0.3 + 1 \times 0.5 \\ = \; & \sum_{(i,j) \in A} c_{ij} f_{ij}/r. \end{aligned}$$

In general, we can prove $\sum_{(i,j)\in A} c_{ij} f_{ij}/r$ represents the average end-to-end link delay of a unicast flow, as given in the following proposition:

Proposition. *Let r be the streaming rate of a unicast session, c_{ij} be the link delay and f_{ij} be the transmission rate on link (i,j), $\forall\ (i,j) \in A$. $\sum_{(i,j)\in A} c_{ij} f_{ij}/r$ represents the average end-to-end link delay of this unicast flow.*

Proof: Let P be the set of paths from the streaming server to the receiver in the session. Let $f^{(p)}$ be the rate of the fractional flow going along path $p \in P$. The average end-to-end latency at the receiver is

$$\sum_{p \in P} \frac{f^{(p)}}{r} \Big(\sum_{(i,j):(i,j) \text{ on } p} c_{ij} \Big) \;=\; \frac{1}{r} \sum_{(i,j)\in A} c_{ij} \Big(\sum_{p:(i,j) \text{ on } p} f^{(p)} \Big) = \frac{1}{r} \sum_{(i,j)\in A} c_{ij} f_{ij}.$$

□

Next, we formulate a linear program to achieve minimum-delay unicast streaming. Let u_{ij} be the capacity of link (i,j). Omitting constant r, we use $\sum_{(i,j)\in A} c_{ij} f_{ij}$ to represent

the average end-to-end link latency of the unicast flow and derive

$$\min \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{3.1}$$

subject to

$$\begin{aligned} \textstyle\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b_i, \quad & \forall\ i \in N, \\ 0 \le f_{ij} \le u_{ij}, \quad & \forall\ (i,j) \in A, \end{aligned}$$

where

$$b_i = \begin{cases} r & \text{if } i = S, \\ -r & \text{if } i = t, \\ 0 & \text{otherwise.} \end{cases}$$

We call the optimal unicast flow decided by this linear program a *minimum-delay flow*. Such a minimum-delay flow is useful in modeling minimum delay multicast streaming in a P2P network, as a minimum-delay multicast streaming flow can be viewed as consisting of multiple minimum-delay flows from the server to each of the receivers. Here we make use of the concept of *conceptual flow* introduced in [58]. A multicast flow is conceptually composed of multiple unicast flows from the sender to all receivers. These unicast conceptual flows co-exist in the network without contending for link capacities, and the multicast flow rate on a link is the maximum of the rates of all the conceptual flows going along this link. For the example shown in Fig. 3.1, the multicast streaming flow from S to t_1, t_2, t_3 and t_4 can be understood as consisting of four conceptual flows from S to each of the receivers. When each conceptual flow is a minimum-delay flow, the end-to-end delays of the multicast session are minimized. Based on this notion, we proceed to formulate the optimal rate allocation problem for multicast P2P streaming.

3.1.2 LP for multicast P2P streaming

Our linear optimization model aims to minimize the end-to-end link delays from the server to all receivers. Based on the initial mesh topology decided by the neighbor assignment from the bootstrapping service, it optimally allocates the transmission rate on each overlay link to construct a minimum (link) delay *streaming topology*. In our formulation, we consider upload and download capacity constraints at each peer, rather than link capacity constraints. This comes from practical observations that bandwidth bottlenecks usually occur on "last-mile" access links at each of the peers in a P2P network, rather than at the Internet core.

Let r be the end-to-end streaming rate of the session. Let f^t denote the conceptual flow from S to a receiver t, $\forall\ t \in T$. f_{ij}^t denotes the rate of f^t flowing through link (i, j). x_{ij} is the actual multicast streaming rate on link (i, j) and c_{ij} is the delay on link (i, j), $\forall\ (i, j) \in A$. For node i, O_i is its upload capacity and I_i is its download capacity. We assume all these variables are non-negative. The linear program is formulated in Table 3.1.

In $\mathbf{P}$, each conceptual flow f^t is a valid network flow, subject to constraints (3.2)(3.3)(3.4) similar to those in the LP in (3.1). The difference lies in that f_{ij}^t's, $\forall\ t \in T$, are bounded by the transmission rate x_{ij} on link (i, j), while x_{ij}'s are further restricted by upload and download capacities at their incident nodes.

An optimal solution to problem $\mathbf{P}$ provides an optimal rate $f_{ij}^{t\,*}$ for the conceptual flow f^t on link (i, j), $\forall\ (i, j) \in A$. Let z be the optimal multicast streaming flow in the network. We compute the optimal transmission rates as:

$$z_{ij} = \max_{t \in T} f_{ij}^t, \quad \forall\ (i, j) \in A. \tag{3.7}$$

Table 3.1: LP for multicast P2P streaming

P:

$$\min \sum_{t \in T} \sum_{(i,j) \in A} c_{ij} f_{ij}^t$$

subject to

$$\sum_{j:(i,j) \in A} f_{ij}^t - \sum_{j:(j,i) \in A} f_{ji}^t = b_i^t, \qquad \forall\ i \in N,\ \forall\ t \in T \quad (3.2)$$

$$f_{ij}^t \geq 0, \quad \forall\ (i,j) \in A,\ \forall\ t \in T \quad (3.3)$$

$$f_{ij}^t \leq x_{ij}, \quad \forall\ (i,j) \in A,\ \forall\ t \in T \quad (3.4)$$

$$\sum_{j:(i,j) \in A} x_{ij} \leq O_i, \qquad \forall\ i \in N \quad (3.5)$$

$$\sum_{j:(j,i) \in A} x_{ji} \leq I_i, \qquad \forall\ i \in N \quad (3.6)$$

where

$$b_i^t = \begin{cases} r & \text{if } i = S, \\ -r & \text{if } i = t, \\ 0 & \text{otherwise.} \end{cases}$$

Such an optimal rate allocation ($z_{ij},\ \forall\ (i,j) \in A$) guarantees r at all the receivers, and achieves minimal end-to-end link latencies as well. At the same time, it computes an optimal peer selection strategy, *i.e.*, an upstream peer is selected at a receiver if the optimal transmission rate between them is non-zero.

3.2 Distributed Solution

We now design an efficient distributed algorithm to solve the linear program **P**. General LP algorithms, such as the Simplex, Ellipsoid and Interior Point methods, are inherently

centralized and costly, which are not appropriate for our purpose. Our solution is based on the technique of Lagrangian relaxation and subgradient algorithm [18, 75], which can be efficiently implemented in a fully distributed manner.

3.2.1 Lagrangian dualization

We start our solution by relaxing the constraint group (3.4) in **P** to obtain its Lagrange dual. The reason of selecting this set of constraints to relax is that the resulting Lagrangian subproblem can be decomposed into classical LP problems, for each of which efficient algorithms exist. We associate Lagrange multipliers μ_{ij}^t with the constraints in (3.4) and modify the objective function as:

$$\sum_{t\in T}\sum_{(i,j)\in A} c_{ij}f_{ij}^t + \sum_{t\in T}\sum_{(i,j)\in A}\mu_{ij}^t(f_{ij}^t - x_{ij}) = \sum_{t\in T}\sum_{(i,j)\in A}(c_{ij}+\mu_{ij}^t)f_{ij}^t - \sum_{t\in T}\sum_{(i,j)\in A}\mu_{ij}^t x_{ij}.$$

We then derive the Lagrange dual of the primal problem **P**:

DP:

$$\max_{\mu\geq 0} L(\mu)$$

where

$$L(\mu) = \min_{P}\sum_{t\in T}\sum_{(i,j)\in A}(c_{ij}+\mu_{ij}^t)f_{ij}^t - \sum_{t\in T}\sum_{(i,j)\in A}\mu_{ij}^t x_{ij} \tag{3.8}$$

and the polytope P is defined by the following constraints:

$$\begin{aligned} \textstyle\sum_{j:(i,j)\in A} f_{ij}^t - \sum_{j:(j,i)\in A} f_{ji}^t = b_i^t, \quad & \forall\ i\in N,\ \forall\ t\in T, \\ f_{ij}^t \geq 0, \quad & \forall\ (i,j)\in A,\ \forall\ t\in T, \end{aligned}$$

$$\sum_{j:(i,j)\in A} x_{ij} \leq O_i, \qquad \forall\ i \in N,$$
$$\sum_{j:(j,i)\in A} x_{ji} \leq I_i, \qquad \forall\ i \in N.$$

Here, the Lagrange multiplier μ_{ij}^t can be understood as the link price on link (i, j) for the conceptual flow from server S to receiver t. Such interpretation should be clear as we come to the adjustment of μ_{ij}^t in the subgradient algorithm.

We observe that the Lagrangian subproblem in Eq. (3.8) can be decomposed into a maximization problem in (3.9),

$$\max \sum_{t\in T} \sum_{(i,j)\in A} \mu_{ij}^t x_{ij} \tag{3.9}$$

subject to

$$\sum_{j:(i,j)\in A} x_{ij} \leq O_i, \quad \forall\ i \in N,$$
$$\sum_{j:(j,i)\in A} x_{ji} \leq I_i, \quad \forall\ i \in N,$$

and multiple minimization problems in (3.10), each for one $t \in T$,

$$\min \sum_{(i,j)\in A} (c_{ij} + \mu_{ij}^t) f_{ij}^t \tag{3.10}$$

subject to

$$\sum_{j:(i,j)\in A} f_{ij}^t - \sum_{j:(j,i)\in A} f_{ji}^t = b_i^t, \quad \forall\ i \in N,$$
$$f_{ij}^t \geq 0, \qquad \forall\ (i, j) \in A.$$

We notice that the maximization problem in (3.9) is an inequality constrained transportation problem, which can be solved in polynomial time by distributed algorithms, *e.g.*, the Auction algorithm [19]. Each minimization problem in (3.10) is essentially a shortest path problem, which finds the shortest path to deliver a conceptual flow of rate r from server S to a receiver t. For the classical shortest path problem, efficient distributed algorithms exist, *e.g.*, Bellman-Ford algorithm, label-correcting algorithms [14] and relaxation algorithms [21]. As the algorithms are all essentially the same as Bellman-Ford algorithm, we employ the distributed Bellman-Ford algorithm [20, 21] as our solution.

3.2.2 Subgradient algorithm

We now describe the subgradient algorithm, applied to solve the Lagrange dual problem **DP**. The algorithm starts with a set of initial non-negative Lagrange multiplier values $\mu_{ij}^t[0],\ \forall\ (i,j) \in A,\ \forall\ t \in T$. At the k^{th} iteration, given current Lagrange multiplier values $\mu_{ij}^t[k]$, we solve the transportation problem in (3.9) and the shortest path problems in (3.10) to obtain new primal variable values $x_{ij}[k]$ and $f_{ij}^t[k]$. Then, the Lagrange multipliers are updated by

$$\mu_{ij}^t[k+1] = \max(0, \mu_{ij}^t[k] + \theta[k](f_{ij}^t[k] - x_{ij}[k])),\ \forall\ (i,j) \in A,\ \forall\ t \in T, \tag{3.11}$$

where θ is a prescribed sequence of step sizes that decides the convergence and the convergence speed of the subgradient algorithm. When θ satisfies the following conditions, the algorithm is guaranteed to converge to $\mu^* = (\mu_{ij}^t,\ \forall\ (i,j) \in A,\ \forall\ t \in T)$, an optimal solution of **DP**:

$$\theta[k] > 0, \lim_{k\to\infty}\theta[k] = 0,\ \text{and}\ \sum_{k=1}^{\infty}\theta[k] = \infty.$$

Eq. (3.11) can be understood as the adjustment of link price for each conceptual flow on each link. If the rate of the conceptual flow exceeds the transmission rate on the link, (3.4) is violated, so the link price is raised. Otherwise, the link price is reduced.

For linear programs, the primal variable values derived by solving the Lagrangian subproblem in Eq. (3.8) at μ^* are not necessarily an optimal solution to the primal problem **P**, and even not a feasible solution to it [73]. Therefore, we use the algorithm introduced by Sherali *et al.* [73] to recover the optimal primal values ${f_{ij}^t}^*$. At the k^{th} iteration of the subgradient algorithm, we also compose a primal iterate $\widehat{f_{ij}^t}[k]$ via

$$\widehat{f_{ij}^t}[k] = \sum_{h=1}^{k} \lambda_h^k f_{ij}^t[h], \ \forall \ (i,j) \in A, \ \forall \ t \in T \tag{3.12}$$

where $\sum_{h=1}^{k} \lambda_h^k = 1$ and $\lambda_h^k \geq 0$, for $h = 1, \ldots, k$. Thus, $\widehat{f_{ij}^t}[k]$ is a convex combination of the primal values obtained in the earlier iterations.

In our algorithm, we choose the step length sequence $\theta[k] = a/(b + ck), \ \forall \ k, a > 0, b \geq 0, c > 0$, and convex combination weights $\lambda_h^k = 1/k, \ \forall \ h = 1, \ldots, k, \ \forall \ k$. These guarantee the convergence of our subgradient algorithm; they also guarantee that any accumulation point $\widehat{f}^*$ of the sequence $\{\widehat{f}[k]\}$ generated via (3.12) is an optimal solution to the primal problem **P** [73]. We can thus calculate $\widehat{f_{ij}^t}[k]$ by

$$\widehat{f_{ij}^t}[k] \ = \ \sum_{h=1}^{k} \frac{1}{k} f_{ij}^t[h] = \frac{k-1}{k} \sum_{h=1}^{k-1} \frac{1}{k-1} f_{ij}^t[h] + \frac{1}{k} f_{ij}^t[k] = \frac{k-1}{k} \widehat{f_{ij}^t}[k-1] + \frac{1}{k} f_{ij}^t[k].$$

3.2.3 Distributed algorithm

Based on the subgradient algorithm, we now design a distributed algorithm to solve **P**, given in Table 3.2. In practice, the algorithm to be executed on a link (i, j) is delegated

by receiver j. Therefore, the algorithm is executed in a fully decentralized manner, in that each peer is only responsible for computation tasks on all its incoming links with only local information, *e.g.*, knowledge of neighbor nodes, delay on its adjacent links, etc.

Table 3.2: The distributed optimal rate allocation algorithm

1. Choose initial Lagrange multiplier values $\mu_{ij}^t[0]$, $\forall\ (i,j) \in A$, $\forall\ t \in T$.

2. Repeat the following iteration until the sequence $\{\mu[k]\}$ converges to μ^* and the sequence $\{\widehat{f}[k]\}$ converges to $\widehat{f}^*$:

   At times $k = 1, 2, \ldots,\ \forall\ (i,j) \in A,\ \forall\ t \in T$

   1) Compute $x_{ij}[k]$ by the distributed auction algorithm;

   2) Compute $f_{ij}^t[k]$ by the distributed Bellman-Ford algorithm;

   3) Compute $\widehat{f_{ij}^t}[k] = \frac{k-1}{k}\widehat{f_{ij}^t}[k-1] + \frac{1}{k}f_{ij}^t[k]$;

   4) Update Lagrange multiplier $\mu_{ij}^t[k+1] = \max(0, \mu_{ij}^t[k] + \theta[k](f_{ij}^t[k] - x_{ij}[k]))$, where $\theta[k] = a/(b+ck)$.

3. Compute the optimal transmission rate $z_{ij} = \max_{t\in T} \widehat{f_{ij}^t}^*$, $\forall\ (i,j) \in A$.

3.3 Handling Peer Dynamics

In P2P streaming, peers may arbitrarily join a streaming session at any time, and may depart or fail unexpectedly. We design the distributed optimal rate allocation algorithm to be invoked and executed in a dynamic manner, and the peer connectivity is reconfigured with adjusted rates with peer dynamics.

Peer joins

In our design, a new peer is admitted into a streaming session only if its download capacity is no lower than the required streaming rate r. It then immediately starts streaming

with the available upload capacities acquired from its upstream peers, assigned by the bootstrapping service. Meanwhile, it sends a request to the streaming server, asking for computation of new optimal rate allocation on the links.

Peer departures and failures

During streaming, when a peer detects the failure or departure of an upstream peer, it attempts to acquire more upload bandwidth from its remaining upstream peers. Only when the peer fails to acquire the required streaming rate, it sends a re-calculation request to the server for the new optimal rate allocation.

At the server, when the number of received re-computation requests exceeds a certain threshold, the server broadcasts such a request, such that all peers activate a new round of distributed algorithm execution, while continuing with their own streaming at the original optimal rates. Note that in such a dynamic environment, a new round of algorithm execution always starts from the previously converged optimal rates, rather than from the very beginning when all the values are zero, thus expediting its convergence. The peers adjust their rates to the new optimal values after the rate allocation converges.

3.4 Performance Evaluation

The focus of our evaluation using extensive simulations is to investigate the performance of our optimal peer selection algorithm over realistic network topologies, with respect to convergence, optimality, and failure resilience. We generate random networks with power-law degree distributions with the BRITE topology generator [64]. We simulate a live streaming session of a high-quality 300 Kbps multimedia bitstream from a streaming

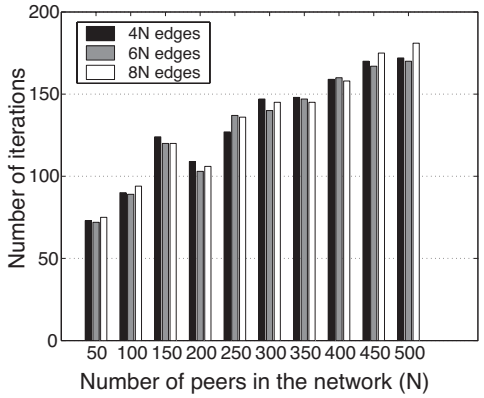


Figure 3.3: Convergence speed in random networks.

server with 10 Mbps of upload capacity. There are two classes of receivers: ADSL/cable modem peers and Ethernet peers. In our general setting, ADSL/cable modem peers take 70% of the total population with $1.5 - 4.5$ Mbps of download capacity and $0.6 - 0.9$ Mbps of upload capacity, and Ethernet peers take the other 30% with both upload and download capacities in the range of $8 - 12$ Mbps. We use link delays generated with BRITE as well.

3.4.1 Convergence speed

We first investigate the convergence speed of our distributed algorithm to obtain the optimal streaming topology. The result is shown in Fig. 3.3. We compare the convergence speed in networks of different *network sizes* (numbers of peers in the network) and different *edge densities* (the ratio of the number of edges to the number of peers in the network). We can see that it takes around 70 iterations to converge to optimality in a network of 50 peers, and this number increases slowly to about 170 for a network of 500 peers. However, the convergence speed remains approximately the same in a fixed-sized network with different edge densities. Therefore, the slow increase of iteration numbers

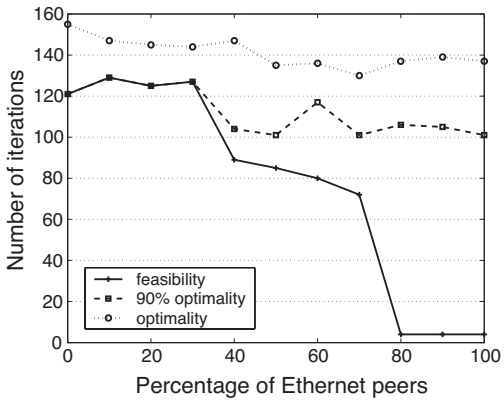


Figure 3.4: Convergence speed to feasibility, 90%-optimality and optimality in random networks of 300 peers and 2400 edges.

with network sizes does not affect the scalability of our algorithm.

We further compare the convergence speeds of our algorithm to the first primal feasible solution, to the feasible solution which achieves 90% optimality as to the value of the objective function, and to the optimal solution. From Fig. 3.4, we observe that the convergence speed to the first primal feasible solution is usually much faster than the convergence to optimality. It can also be seen that the number of iterations needed to converge to feasibility drops quickly with the increase of the percentage of Ethernet peers in the network, which bring more abundant upload capacities. Furthermore, in order to converge to the feasible solution which achieves 90% optimality, the algorithm takes only 75% of the number of iterations required for convergence to the optimal solution. Therefore, in practice, we can obtain a feasible solution to a certain degree of the optimality in a much shorter time, when it is not always necessary to achieve the optimal solution in a realistic streaming system.

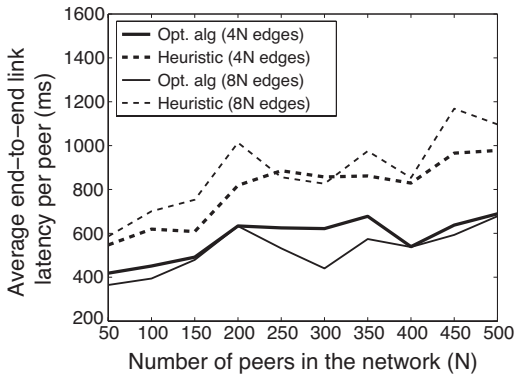


Figure 3.5: Average end-to-end latency: a comparison between optimal peer selection algorithm and a peer selection heuristic.

3.4.2 Optimality

We next compare our optimal peer selection algorithm with a commonly used peer selection heuristic [56, 88]. In the heuristic, a receiver distributes the streaming rates among its upstream peers in proportion to their upload capacities. We compare the end-to-end link latencies at receivers in the resulting streaming topologies. The end-to-end latency at each receiver is calculated as the weighted average of the delays of flows from all its upstream peers, and the weight for each flow is the portion of the assigned streaming rate from the upstream peer in the aggregate streaming rate.

The results illustrated in Fig. 3.5 meet our expectations. In networks of different network sizes and edge densities, our end-to-end latency minimization algorithm is able to achieve much lower latencies than the heuristic, which does not take link delay into consideration. We further notice that the denser the network is, the higher the average end-to-end latency is by the heuristic. In contrast, our optimal algorithm achieves lower latencies in denser networks. When the edge density is $4N$ in a network of N peers, the average end-to-end latency of the heuristic is about 1.5 times higher than that of our optimal algorithm, while this ratio becomes 2 in a network with $8N$ edges. For such

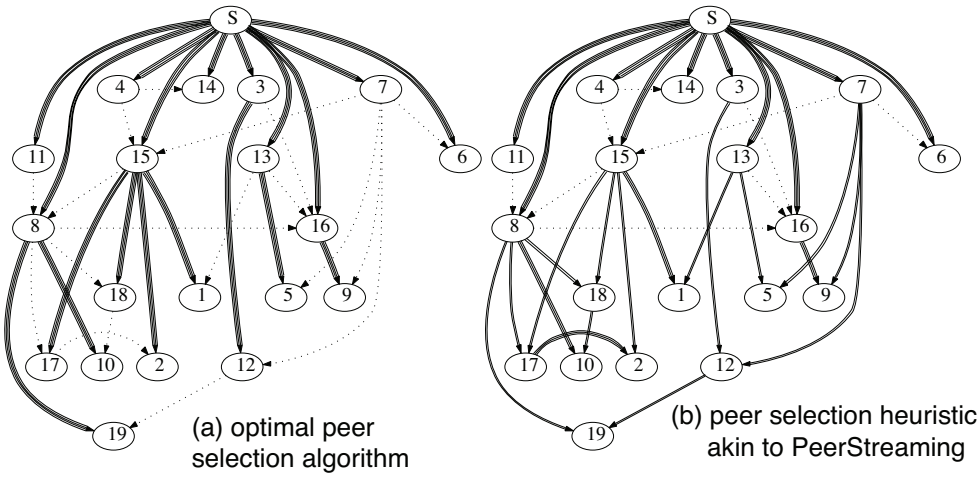


Figure 3.6: P2P streaming topologies of 20 peers: a comparison.

an achievement of lower latencies in denser networks with our algorithm, we believe the reason is that there are more choices of upstream peers in a denser network and our algorithm can always find the best set of upstream peers on low delay paths. Thus, in realistic P2P streaming networks with high edge densities, the advantage of our algorithm is more evident over the commonly used heuristic.

The streaming topologies shown in Fig. 3.6(a) and Fig. 3.6(b) further illustrate the superiority of our optimal algorithm. In these graphs, distances between pairs of peers represent latencies, and the widths of edges show the streaming rates along them. The dotted lines represent links that are not used in the resulting streaming topologies. It can be seen that by our optimal peer selection, receivers are streaming from the best upstream peers with minimal end-to-end latencies, while with the peer selection heuristic, peers simply distribute their download rates among the upstream peers, which may lead to large end-to-end latencies.

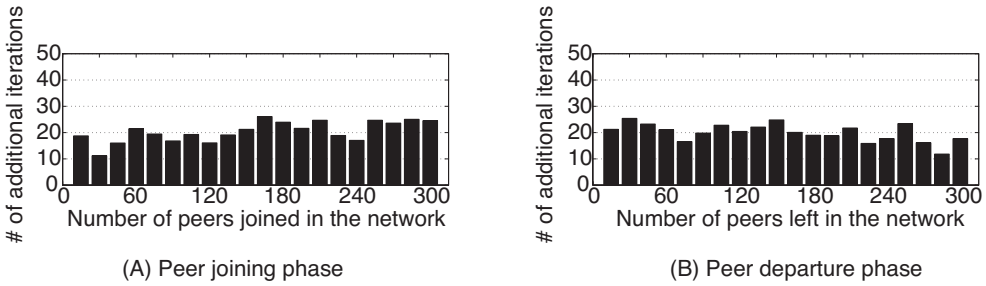


Figure 3.7: Convergence speed in a dynamic network with up to 300 peers.

3.4.3 Adaptation to dynamics

To investigate the practicality of our algorithm, we examine the convergence speed of the optimization algorithm in dynamic networks. In this experiment, during a 45-minute streaming session, 300 peers sequentially join the session in the first 20 minutes, and then start to depart from 25 minutes onwards. The distributed optimal rate allocation algorithm is invoked about every 15 peer joins or departures, and always runs from the previous optimal flow rates, following the dynamic execution method described in Sec. 3.3.

The number of additional iterations needed to converge to the new optimal rates in both the peer joining phase and departure phase are illustrated in Fig. 3.7. The results reveal that the convergence to new optimal rates in such dynamic scenarios is much faster, as compared to running from the very beginning in the case of static networks of the same sizes. Independent of the current network size, the algorithm always takes less than 30 iterations to converge.

We note that although this is a specially designed dynamic case, it reflects the capability of the optimization algorithm to converge promptly from one optimum to another in practical dynamic scenarios. In a realistic P2P streaming network, peer joins and departures may occur concurrently and consistently during the entire streaming session.

In this case, our algorithm always improves the rate allocation towards optimality in the current network and can converge quickly as long as there exists a stable operating point in the dynamic overlay.

3.5 Summary

In this chapter, we design an efficient distributed algorithm for optimal peer selection and streaming rate allocation in a mesh-based P2P live streaming session. We formulate the problem as a linear optimization problem, which optimizes bandwidth utilization towards minimized end-to-end latencies, and develop a fully decentralized algorithm to efficiently compute the optimal streaming rates on the P2P links. With extensive simulations, we show how this optimization algorithm can be practically carried out in realistic P2P networks to guarantee the optimality of the streaming topologies.

Chapter 4

Meeting Bandwidth Demand in Practical Streaming

The most fundamental requirement of P2P streaming is to achieve and maintain a specific streaming playback rate at each participating peer, in order to guarantee the smooth playback of the media. For example, with the H.264 codec, a Standard-Definition stream demands approximately 800 Kbps, while 480p (848×480 pixels) High-Definition media stream using the H.264 codec requires 1700 Kbps. In the previous chapter, we have designed an optimal streaming rate allocation algorithm, that achieves the required streaming rate at the peers, given *a prior* knowledge of their bandwidth capacities. In this chapter, we continue to investigate: How can such a fundamental streaming rate requirement be practically satisfied in realistic P2P networks, without any *a prior* knowledge of bandwidth availabilities on the peer nodes or overlay links?

Such a practical streaming rate satisfaction problem is not explicitly nor well addressed in existing P2P streaming solutions [40, 50, 87, 88]. In the recent work of Chainsaw [70], peers are sending packets as their bandwidths allow, but it is not specified how

the sending rates towards different neighbors are to be regulated, such that a required streaming playback rate can be achieved. In PRIME [62], Magharei *et al.* suggest that each P2P connection in the mesh streaming overlay should have roughly the same bandwidth in order to maximize the utilization of peer access link bandwidth, but have not emphasized on how to achieve this in practice.

We identify that whether or not the streaming playback rate requirement, hereafter referred to as streaming bandwidth *demand*, can be achieved at the peers depends on three constraints. First, the last-mile download capacity must exceed the streaming rate. We typically assume that this constraint is always satisfied, as otherwise the required streaming rate cannot be achieved with any solution. It is most likely the case in reality, as peers without sufficient last-mile download capacities would soon leave the session, and join another session to download the media encoded at lower bit rates. Second, as the peer in a mesh network is served by multiple upstream peers, the last-mile upload capacities of these upstream peers are limited. Finally, the available bandwidth on each overlay link between two peers is limited, subject to link capacity and cross traffic in the Internet core.

In essence, the decisive factor of meeting the P2P streaming bandwidth demand in a streaming session is the *bandwidth supply* from either dedicated streaming servers or uploading peers. Ideally, when the total bandwidth supply is abundant, a peer can easily contact new streaming servers or peers when its demand cannot be met at any time. However, such a simple solution does not work effectively at all when there exist very limited bandwidth supplies to meet the demand. Such a microeconomic phenomenon of tight supply-demand relationships is usually the norm in realistic P2P streaming networks, especially during flash crowd scenarios when a limited pool of streaming servers

scrambles to meet the demand of a P2P session that scales up in a short period of time.

In this chapter, we seek to design a new bandwidth allocation algorithm that takes advantage of existing bandwidth supplies in the most efficient manner in scenarios with tight supply-demand relationships. Our algorithm dynamically adjusts the bandwidth utilization on each overlay link, so that the streaming bandwidth demand is maximally guaranteed on each participating peer in the session, even with very limited bandwidth supplies.

The new algorithm we propose enjoys the following salient advantages. *First*, although the rates allocated are subject to the capacity constraints at the edge and on the overlay links, our new algorithm does not need knowledge of these capacity constraints. *Second*, our algorithm is fully decentralized, and can thus be realistically implemented. To design such an algorithm, we formulate the streaming rate satisfaction problem in P2P streaming as a *feasibility* problem, and propose a simple algorithm to find its solution. We discuss the implementation and prove the convergence of the algorithm in synchronous and asynchronous environments. *Third*, we show that even in cases of persistent peer dynamics and network changes, the algorithm is still guaranteed to pursue the required streaming rate at each peer. *Finally*, our solution is across-the-board and not specific to any particular mesh-based topology construction mechanism, P2P streaming protocol, or media codec, as the problem of bandwidth allocation to guarantee smooth streaming playback is fundamental and widely applicable to any P2P streaming algorithm. In addition, bandwidth allocation may also be realistically implemented in any P2P streaming protocol, by using per-link bandwidth shaping mechanisms at peers (usually based on UDP as the transport protocol).

4.1 Motivation and Problem Formulation

Consider a mesh-based P2P streaming session, $\mathcal{G} = (\mathcal{S}, \mathcal{N}, \mathcal{A})$, where $\mathcal{S}$ is the set of streaming servers, $\mathcal{N}$ is the set of participating peers, and $\mathcal{A}$ is the set of directed overlay links. We assume such a mesh topology is constructed and maintained with a certain topology construction protocol, *e.g.*, the random mesh construction employed by most current-generation P2P streaming applications. The overlay links among peers in the topology are established based on their media content availability during streaming, *i.e.*, a peer streams from one or more upstream peers, which can provide it with media blocks it requires, and further serves one or more downstream peers with the media streams.

In this chapter, we study both the "demand" and the "supply" of bandwidth based on the given topology. On the side of *demand* of bandwidth, a streaming playback rate $\mathcal{R}$ is strictly required at each participating peer to guarantee smooth playback, *i.e.*, media content should be downloaded at an aggregated bandwidth of no lower than $\mathcal{R}$ at the peer. On the side of *supply* of bandwidth, however, we need to consider both peer upload bandwidth and overlay link bandwidth constraints, without *a priori* knowledge on either. How do we meticulously allocate the supply of bandwidth so that the streaming playback rate $\mathcal{R}$ — the "demand" in each session — can be satisfied at all times? This problem is henceforth referred to as the *streaming rate satisfaction* problem, as we seek to design a practical and decentralized algorithm to address such a challenge. While our focus in the chapter is not on the specific P2P topology construction protocol, we will discuss the interactive play between our bandwidth allocation and topology construction in P2P streaming in Sec. 4.3.

Since we do not need to consume *more* bandwidth than the required streaming playback rate at each peer, we believe it is practical to formulate such a streaming rate

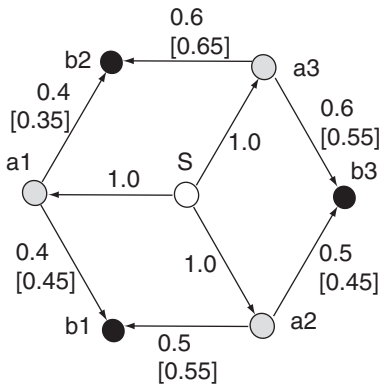


Figure 4.1: Infeasible bandwidth allocation with a naive protocol: an example.

satisfaction problem as a *feasibility* problem, and aim to find a *feasible* bandwidth allocation solution that guarantees a streaming rate of no lower than $\mathcal{R}$ at each peer.

One may wonder why a naive bandwidth allocation may fail to satisfy the streaming bandwidth demand at all peers. We show this with an example in Fig. 4.1, which implies that an explicit bandwidth allocation algorithm is required.

In this example P2P streaming network, the required streaming rate is $\mathcal{R} = 1$ Mbps. Peers a_1, a_2, and a_3 directly stream at this rate from server s, which has 3 Mbps of upload capacity, and then serve b_1, b_2, and b_3. Assume that bandwidth bottlenecks occur at the upload links of the three a peers, with upload capacities of 0.8, 1.0 and 1.2 (in Mbps), respectively. With a naive even bandwidth allocation method, their upload capacities are evenly shared among their respective downstream peers, and the allocated bandwidths are labeled on the links (numbers outside the brackets). Such a bandwidth allocation outcome is infeasible (*i.e.,* streaming bandwidth demand is not satisfied at all peers), as peer b_1 only obtains an aggregate bandwidth of 0.9 Mbps, while b_3 is allocated more than 1 Mbps.

When the required streaming bandwidth is not successfully achieved, the common

practice with existing P2P protocols is to find new upstream peers, or to start a new connection from the streaming server, which may well fail to locate available bandwidth when there is a tight supply-demand relationship of bandwidth in the network. It is important to note, however, that if bandwidths are explicitly *reallocated* based on the current topology, a feasible solution that satisfies all the peers can be achieved, as shown in brackets in our example. After such a "reorganizing" process, the "supply" of bandwidth is maximally utilized, the need to find new upstream peers is eliminated, and server bandwidth costs are minimized.

In order to design a practical algorithm to achieve such feasible rate allocation solutions in a particular P2P mesh topology, we first formally formulate the problem. Let x_{ij} denote the allocated transmission rate on the overlay link $(i,j),\ \forall\ (i,j) \in \mathcal{A}$. In practical P2P systems, such overlay link rates are restricted by the capacities of last-mile access links of the peers, and affected by the cross traffic sharing the same underlying IP network. Let $\mathcal{C}_{ij}$ denote the currently available bandwidth along overlay link (i,j), subject to the cross traffic. Let $\mathcal{O}_i$ denote the upload capacity at peer $i,\ \forall\ i \in \mathcal{S} \cup \mathcal{N}$. The feasibility problem is formally defined by the following set of linear rate and capacity constraints:

LC:

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} \geq \mathcal{R}, \qquad \forall\ j \in \mathcal{N}, \tag{4.1}$$

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij} \leq \mathcal{O}_i, \quad \forall\ i \in \mathcal{S} \cup \mathcal{N}, \tag{4.2}$$

$$x_{ij} \leq \mathcal{C}_{ij}, \quad \forall\ (i,j) \in \mathcal{A}. \tag{4.3}$$

Let $x = (x_{ij}, (i,j) \in \mathcal{A})$ be the $|\mathcal{A}|$-dimensional vector of allocated link bandwidths.

Let X_R be the region defined by the streaming rate constraints in (4.1), *i.e.*, $X_R = \{x : \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \geq \mathcal{R},\ \forall\ j \in \mathcal{N}\}$. Let X_C be the region defined by the capacity constraints in (4.2) and (4.3), *i.e.*, $X_C = \{x : \sum_{j:(i,j)\in\mathcal{A}} x_{ij} \leq \mathcal{O}_i,\ \forall\ i \in \mathcal{S}\cup\mathcal{N}, x_{ij} \leq \mathcal{C}_{ij},\ \forall\ (i,j) \in \mathcal{A}\}$. A solution to this feasibility problem represents a feasible bandwidth allocation scheme, expressed as $x \in X_R \cap X_C$.

4.2 The Synchronous Case

We are now ready to propose our iterative algorithm to solve the feasibility problem **LC**. We first show that it can be readily implemented in a fully decentralized fashion, and analyze its convergence in the synchronous case.

4.2.1 An iterative algorithm

Inspired by the iterative optimization algorithm proposed by Kar *et al.* [45], we design a simple iterative algorithm to derive a feasible solution satisfying all the constraints in the problem **LC**.

Let $x_{ij}^{(n)}$ be the allocated rate on link $(i,j),\ \forall\ (i,j) \in \mathcal{A}$, at the n^{th} step. Let

$$\lambda_j^{(n)} = \max(0, \mathcal{R} - \sum_{i:(i,j)\in\mathcal{A}} x_{ij}^{(n)}),$$

and

$$e_{ij}^{(n)} = \begin{cases} 1 & \text{if } \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(n)} > \mathcal{O}_i, \\ & \quad \text{or } x_{ij}^{(n)} > \mathcal{C}_{ij}, \\ 0 & \text{otherwise.} \end{cases}$$

We update $x_{ij}^{(n)}$ by:

$$x_{ij}^{(n+1)} = \begin{cases} x_{ij}^{(n)} & \text{if } \lambda_j^{(n)} = 0, e_{ij}^{(n)} = 0, \\ x_{ij}^{(n)} + \alpha_n \lambda_j^{(n)} & \text{if } \lambda_j^{(n)} > 0, e_{ij}^{(n)} = 0, \\ x_{ij}^{(n)} - \beta_n e_{ij}^{(n)} & \text{if } \lambda_j^{(n)} = 0, e_{ij}^{(n)} > 0, \\ x_{ij}^{(n)} + \alpha_n \lambda_j^{(n)} - \beta_n e_{ij}^{(n)} & \text{if } \lambda_j^{(n)} > 0, e_{ij}^{(n)} > 0, \end{cases} \tag{4.4}$$

where α_n and β_n are two sequences with the following properties:

$$\lim_{n\to\infty} \alpha_n = 0, \sum_{n=1}^{\infty} \alpha_n = \infty, \lim_{n\to\infty} \beta_n = 0, \sum_{n=1}^{\infty} \beta_n = \infty, \lim_{n\to\infty} \frac{\alpha_n}{\beta_n} = 0. \tag{4.5}$$

For example, the sequences $\alpha_n = \frac{a}{n}$ and $\beta_n = \frac{b}{\sqrt{n}}$, where a, b are positive constants, satisfy the above properties.

In each step n, $\lambda_j^{(n)}$ represents how much more bandwidth peer j needs to acquire in order to achieve the required streaming rate $\mathcal{R}$. $e_{ij}^{(n)}$ can be understood as a *binary* indicator of insufficient bandwidth, showing whether available bandwidth is exceeded along overlay link (i, j): either the upload capacity of peer i is exceeded, or $x_{ij}^{(n)}$ goes beyond the available bandwidth on overlay link (i, j).

The intuition behind the updates in (4.4) is as follows: Whenever an overlay link does not have sufficient bandwidth, the allocated rate along the link is reduced; whenever the aggregate rate on the download links of a peer falls below $\mathcal{R}$, the allocated link bandwidths are increased, according to how much the aggregate rate deviates from $\mathcal{R}$. α_n and β_n denote the step lengths of the updates. The increment step length α_n is much

smaller than the decrement step length β_n for sufficiently large n, and both of them are diminishing. These are important properties to guarantee the convergence of the algorithm to a feasible solution of **LC**, as will be used in our convergence analysis.

4.2.2 Practical implementation

Our iterative algorithm can be readily implemented in a fully distributed fashion. We first study its decentralized implementation in the synchronous case, where updates are synchronized to occur at times $n = 1, 2, \ldots$. In the subsequent section, we will show that, with minor modifications, the implementation can also achieve feasible bandwidth allocation in asynchronous and dynamic environments.

In the synchronous case, the allocated bandwidth on a link (i, j) is adjusted at the downstream peer j during the actual streaming process. The media streams are transmitted from upstream peers at the allocated transmission rates, x_{ij}'s, using a bandwidth shaping mechanism.

In our implementation, at times $n = 1, 2, \ldots$, peer j calculates $\lambda_j^{(n)}$ based on the discrepancy between $\mathcal{R}$ and the currently allocated rates on its download links, *i.e.*, $\lambda_j^{(n)} = \max(0, \mathcal{R} - \sum_{i:(i,j)\in\mathcal{A}} x_{ij}^{(n)})$. If $\lambda_j^{(n)} > 0$, it increases the allocated rates by $x_{ij}^{(n)'} = x_{ij}^{(n)} + \alpha_n \lambda_j^{(n)}, \ \forall\, i : (i, j) \in \mathcal{A}$; otherwise, it sets $x_{ij}^{(n)'} = x_{ij}^{(n)}$. Meanwhile, peer j estimates the actually achieved receiving rate $y_{ij}^{(n)}$ from each of its upstream peers, by dividing the number of bytes received on each link in a time interval by the interval length. It sets $e_{ij}^{(n)} = 1$ if the actual receiving rate is lower than the allocated rate on the link, *i.e.*, $y_{ij}^{(n)} < x_{ij}^{(n)}$, or sets $e_{ij}^{(n)} = 0$ otherwise. It then proceeds to update the allocated rates again by $x_{ij}^{(n+1)} = x_{ij}^{(n)'} - \beta_n e_{ij}^{(n)}, \ \forall\, i : (i, j) \in \mathcal{A}$, and requests these new transmission rates from its respective upstream peers. After an upstream peer receives the new requested

rates $x_{ij}^{(n+1)}$ from all its downstream peers, it adjusts its sending rates to the new values.

We note that our implementation does *not* depend on any *a priori* knowledge of peer upload and overlay link bandwidth, nor any feedback from IP-layer routers. In our implementation, the value of *insufficient bandwidth indicator* $e_{ij}^{(n)}$ on each link (i, j) is inferred by comparing the allocated transmission rate from upstream peer i with the achieved receiving rate at downstream peer j during the streaming process. The rationale behind this is that when an allocated transmission rate is larger than a respective receiving rate, bandwidth insufficiency is implied either at the upstream peer or on the overlay link, *i.e.*, $e_{ij}^{(n)} = 1$; otherwise, no bandwidth limit is exceeded along link (i, j), and $e_{ij}^{(n)} = 0$. In this way, although we formally formulate the problem **LC** with $\mathcal{O}_i$ and $\mathcal{C}_{ij}$ in (4.2) and (4.3), respectively, we do not actually need to perform any bandwidth probing to explicitly derive the values of these bandwidth limits.

4.2.3 Convergence analysis

We now analyze the convergence of our decentralized implementation of the iterative algorithm, in the synchronous case. To facilitate our analysis, we consider the realistic scenario that, if the aggregate requested (sending) rate at an upstream peer i is higher than its upload capacity, the receiving rates at all its downstream peers are lower than their respective requested rate, *i.e.*, each of them is able to detect the bandwidth insufficiency. Our discussion is divided into two cases: (1) a feasible solution exists for **LC** , *i.e.*, $X_R \cap X_C \neq \phi$; and (2) a feasible solution does not exist for **LC**, *i.e.*, $X_R \cap X_C = \phi$.

Theorem 1 shows that, when a feasible solution exists, *i.e.*, there is sufficient bandwidth in the overlay to support all peers at the required streaming rate $\mathcal{R}$, the decentralized implementation of our iterative algorithm converges to such a solution.

Theorem 1. If $X_R \cap X_C \neq \phi$, with iterative updates in (4.4) and diminishing step lengths in (4.5), the sequence $\{x^{(n)}\}$ converges to $\tilde{x}$, a feasible solution of problem **LC**, *i.e.,* $\tilde{x} \in X_R \cap X_C$.

Theorem 2 addresses the second case, when a feasible bandwidth allocation solution does not exist, *i.e.,* the overlay cannot accommodate all the peers at $\mathcal{R}$. Theorem 2 states that, at all the peers, our implementation is able to achieve the maximum throughput supported by the overlay.

Let $\mathcal{R}_{\max}$ be the maximum throughput at the peers that the network can support, *i.e.*, the maximum aggregate streaming bandwidth that each peer can acquire. It implies that there exist feasible solutions to the following problem:

LC':

$$\begin{aligned} \sum_{i:(i,j)\in\mathcal{A}} x_{ij} &\geq \mathcal{R}_{\max}, \qquad \forall\ j \in \mathcal{N}, \\ \sum_{j:(i,j)\in\mathcal{A}} x_{ij} &\leq \mathcal{O}_i, \quad \forall\ i \in \mathcal{S} \cup \mathcal{N}, \\ x_{ij} &\leq \mathcal{C}_{ij}, \quad \forall\ (i,j) \in \mathcal{A}. \end{aligned} \tag{4.6}$$

Theorem 2. If $X_R \cap X_C = \phi$, with iterative updates in (4.4) and diminishing step lengths in (4.5), the sequence $\{x^{(n)}\}$ converges to the feasible region of problem **LC'**, *i.e.*, $\lim_{n\to\infty} \rho(x^{(n)}, X') = 0$, where X' is the region defined by the constraints in **LC'**.

The proofs of these theorems are presented in Sec. 4.6 at the end of the chapter. The key intuition behind the proofs is to show that, based on the diminishing step lengths, in each step of the iterative algorithm, the current bandwidth allocation improves towards a feasible solution of **LC** or approaches the feasible region of **LC'**. Based on these theorems, a corollary follows:

Corollary 1. During the convergence of $\{x^{(n)}\}$, the actually achieved streaming rate at each peer j, i.e., $\sum_{i:(i,j)\in\mathcal{A}} y_{ij}^{(n)}$, is asymptotically increasing and converges to $\mathcal{R}$ if the network can support such rate at each peer, and $\mathcal{R}_{\max}$ otherwise.

During the dynamic process of bandwidth allocation, a peer's achieved streaming rate may temporarily decrease when the allocated rates on its download links decrease due to lack of available bandwidth. However, over time, this achieved streaming rate is asymptotically increasing until it reaches $\min(\mathcal{R}, \mathcal{R}_{\max})$.

4.3 The Asynchronous Case

Granted, while important as a first step in our study, the synchronous case that we have considered is an idealistic view of practical P2P networks. Peers are inherently *asynchronous*, with different processing times and messaging latencies. Fortunately, with minor modifications, we can extend our decentralized implementation to the asynchronous case, with the ability to handle peer and network dynamics.

In an asynchronous overlay, if we execute our decentralized implementation previously proposed for the synchronous case, the step lengths at a certain time t, $\alpha(t)$ and $\beta(t)$, are not identical at all the peers, as peers update the allocated rates at their own paces. However, it is the key to guarantee algorithm convergence by updating bandwidth allocation synchronously with the same step lengths across the network, as used in proofs of Theorem 1 and 2 in Sec. 4.6. Thus the iterative synchronous implementation may fail to converge in the asynchronous case.

Fortunately, we are able to show that, the update process can still be proven to converge to a feasible solution of **LC** in an asynchronous environment, if each peer follows the *synchronous update rule* across its own download and upload links. More

rigorously, on a downstream peer j, all increments of allocated rates on all its download links are performed at the same time t, and use a same diminishing step length $\alpha_j(t)$, *i.e.*, $x_{ij}(t+1) = x_{ij}(t) + \alpha_j(t)\lambda_j(t),\ \forall\ i : (i,j) \in \mathcal{A}$. On the other hand, on an upstream peer i, all decrements of allocated rates on all its upload links are performed at the same time t, and use a same diminishing step length $\beta_i(t)$, *i.e.*, $x_{ij}(t+1) = x_{ij}(t) - \beta_i(t)e_{ij}(t),\ \forall\ j : (i,j) \in \mathcal{A}$.

4.3.1 Practical implementation

We are now ready to present our decentralized implementation in the asynchronous case, and show its convergence to a feasible solution. In the asynchronous implementation, the allocated rate on a link (i,j) is adjusted with the cooperation of both upstream peer i and downstream peer j, *i.e.*, increment at peer j and decrement at peer i. To implement this, rate updates and inferred values of the *insufficient bandwidth indicators* need to be passed between upstream and downstream peers in special protocol messages, referred to as *Rate Update (RU)* messages. These protocol messages are delivered using reliable transport protocols such as TCP.

Our decentralized asynchronous implementation executed at each peer i proceeds as follows.

Initialization:

1. Initialize the set of current upstream peers $\mathcal{U}_i$ and downstream peers $\mathcal{D}_i$, as well as the step counters $n_i = 1$, and $m_i = 1$.

2. For every upstream peer u in $\mathcal{U}_i$:

(2.1) Set $x_{ui} = \mathcal{R}/|\mathcal{U}_i|$ and $e_{ui} = 0$.

(2.2) Send x_{ui} and e_{ui} to peer u with a RU message.

Next, peer i executes the following steps in its dual roles as a downstream peer and an upstream peer, using step counters n_i and m_i, respectively.

As a downstream peer:

1. Receive RU messages from its upstream peers, and estimate the actually achieved receiving rate y_{ui} from each of them. Adjust $\mathcal{U}_i$ if it detects any upstream peer failures.

2. After it has received RU messages from all its existing upstream peers, do the following:

(2.1) Retrieve allocated rates $x_{ui}(t),\ \forall\ u \in \mathcal{U}_i$, from the received RU messages.

(2.2) Compute $\lambda_i(t) = \max(0, \mathcal{R} - \sum_{u:(u,i)\in\mathcal{A}} x_{ui}(t))$.

(2.3) For each upstream peer u:

(2.3.1) If $\lambda_i(t) > 0$, increase the allocated rate by $x_{ui}(t+1) = x_{ui}(t) + \alpha_{n_i}\lambda_i(t)$; otherwise, set $x_{ui}(t+1) = x_{ui}(t)$.

(2.3.2) If $y_{ui} < x_{ui}(t)$, set $e_{ui}(t+1) = 1$; otherwise, set $e_{ui}(t+1) = 0$.

(2.3.3) Send $x_{ui}(t+1)$ and $e_{ui}(t+1)$ to peer u with a RU message.

3. Increment step counter: $n_i = n_i + 1$.

As an upstream peer:

1. Receive RU messages from its downstream peers. Adjust $\mathcal{D}_i$ if it detects any downstream peer failures, or receives RU messages from new downstream peers.

2. After it has received RU messages from all its existing downstream peers, do the following:

For each downstream peer j:

(2.1) Retrieve $e_{ij}(t)$ and $x_{ij}(t)$ from the RU message from peer j.

(2.2) If $e_{ij}(t) = 1$, decrease the allocated rate by $x_{ij}(t+1) = x_{ij}(t) - \beta_{m_i} e_{ij}(t)$; otherwise, set $x_{ij}(t+1) = x_{ij}(t)$.

(2.2) Adjust the sending rate to peer j to $x_{ij}(t+1)$, and send $x_{ij}(t+1)$ in a RU message to peer j.

3. Increment the step counter: $m_i = m_i + 1$.

Theorem 3 shows the convergence of our decentralized asynchronous implementation of the iterative algorithm.

Theorem 3. With our decentralized asynchronous implementation, and under the assumption that both the message passing delay and the time between consecutive updates are finite, the sequence $\{x(t)\}$ (the rate vector at time t) converges to a feasible solution of **LC** if $X_R \cap X_C \neq \phi$, or to the feasible region of **LC'** otherwise.

The proof of Theorem 3 is presented in Sec. 4.6.3. The key to guarantee the convergence is, as pointed out earlier, when a downstream peer i updates the allocated rates on its download links, it increases them altogether with the same diminishing step length α_{n_i}; when an upstream peer i updates the allocated rates on its upload links, it deducts them altogether with the same diminishing step length β_{m_i}. In this case, the bandwidth allocation still improves towards feasibility in each step.

4.3.2 Handling dynamics

We further note that, our asynchronous implementation can maximally guarantee the required streaming bandwidth at each peer, in cases of both peer and network dynamics. To understand such robustness against dynamics, we consider the influence of dynamics *during* and *after* the convergence process.

Dynamics during convergence. Our asynchronous implementation of the algorithm can readily adapt to dynamics introduced before the allocated rates converge. First, when a new peer i joins the streaming session, it is assigned an initial set of upstream peers,

which is decided by the topology construction protocol based on media content availability, and executes the *initialization* phase. After its selected upstream peers receive its RU messages, they include peer i in their respective sets of downstream peers. Thus, peer i can immediately participate in the asynchronous implementation as a downstream peer from the initial step counter $n_i = 1$, while its upstream peers continue their own execution with their current step counter values. Second, in the case of peer failures or departures, after the downstream and upstream peers detect peer i's failure or departure, they simply remove i from their respective sets of upstream or downstream peers and continue with their execution, effectively excluding i from later message exchanges and computation. Finally, our implementation naturally adapts to fluctuating overlay link bandwidth due to the appearing and vanishing of congestion along the links, since our implementation uses binary indicators of insufficient bandwidth e_{ij}, rather than explicit *a priori* knowledge of link bandwidth.

To derive the convergence properties of the asynchronous implementation in dynamic networks, the analysis in Theorem 3 still applies, *i.e.*, we can still show the bandwidth allocation dynamically improves towards feasibility in the current network, and converges to one feasible solution if there exists one in the dynamic network. Formally, we present it as Corollary 2, which can be obtained directly from Theorem 3:

Corollary 2. If dynamics occur during the convergence of our asynchronous implementation of the iterative algorithm, the rate vector $x(t)$ improves towards feasible bandwidth allocation of the current overlay, and converges to a feasible solution whenever there exists one, or maximizes the peer throughput in the dynamic overlay.

Dynamics after convergence. If dynamics occur after allocated rates have converged, the affected peers initiate a new round of protocol execution with reset step counters, in

which the bandwidth allocation continues to improve towards feasibility or throughput maximization.

The handling of the case that a new peer joins the streaming session is similar to that discussed when dynamics occur during convergence, except that all peers now execute the protocol from $n_i = 1$ or $m_i = 1$. In the cases of peer failure or departure, or overlay link bandwidth fluctuations, which have caused the loss of streaming rate at a particular downstream peer, the affected downstream peer will reallocate its bandwidth requests towards its remaining upstream peers, and send out the new RU messages to each of them. In this way, a new round of protocol execution is invoked, and the involved peers reset their step counters and cooperate in the protocol execution from their current bandwidth allocation.

To conclude our analytical discussions of bandwidth allocations in P2P streaming, we reiterate our motivation for this work. We believe that our bandwidth allocation algorithms can help existing topology construction protocols so that they are more effective, especially in the case of tight supply-demand relationships of bandwidth. Once a P2P topology is first constructed using a particular protocol, our bandwidth allocation algorithm can be used to maximally utilize the existing supply of bandwidth in such a topology. In fact, we further point out that our algorithm can directly function as a joint topology construction and bandwidth allocation protocol as well, if complete knowledge at each peer can be assumed (*i.e.*, the input mesh topology $\mathcal{G}$ to our feasibility problem is a complete graph). In this way, a P2P link (i, j) remains in the resulting topology if the computed transmission rate x_{ij} is non-zero.

In a practical large-scale P2P network, it is not realistic to assume that each peer

knows every other peer in the system. Therefore, we have proposed that our protocol can be carried out in a *complementary* fashion with an existing topology construction protocol, which assigns each peer an initial set of a small number of upstream peers (*e.g.,* using random peer selection). Then our bandwidth allocation algorithm can be applied to make the best use of the available bandwidth based on the current peer connectivity. If the required streaming rate is not satisfied on all the peers in the topology, the topology construction protocol can be reactivated, *e.g.*, more upstream peers with available bandwidth are acquired at peers which have not achieved the required streaming rate. After the topology change, our bandwidth allocation algorithm can come into play again, which achieves the highest level of peer streaming rate satisfaction in the new topology. With our bandwidth allocation algorithm, the need for topology reconstruction, *i.e.*, to find new upstream peers and adjust peer connectivity, is minimized, and the feasible streaming rate is achieved quickly at the peers to counter the effects of network dynamics. In essence, such joint functioning of the topology construction protocol and our bandwidth allocation algorithm achieves the joint optimization to derive a desirable mesh streaming structure.

4.4 Performance Evaluation

To evaluate the effectiveness, in particular the dynamic behavior of convergence, we have conducted an in-depth empirical study on our decentralized asynchronous implementation proposed in Sec. 4.3 with C++-based simulations. We choose to simulate our implementation in the asynchronous case due to its practicality in realistic P2P topologies. The first several proof-of-concept experiments use small example topologies, akin to the concept of "unit testing" in software development.

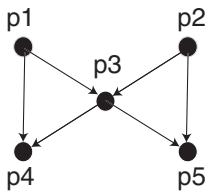


Figure 4.2: An example P2P streaming topology.

Experiment A (behavior of convergence): We first illustrate how the implementation converges in an example topology in Fig. 4.2. We simulate a streaming session with $\mathcal{R} = 800$ Kbps. In this network, peers p_1 and p_2 directly stream from the server (not shown in Fig. 4.2), and serve as "mini-sources" for p_3, p_4 and p_5. The rate vector is $x = (x_{13}, x_{23}, x_{14}, x_{34}, x_{25}, x_{35})$. End-to-end delays on overlay links are $(30, 50, 60, 100, 80, 120)$ (in milliseconds), respectively. We investigate both feasible and infeasible cases.

The feasible case: The overlay topology is able to support p_3, p_4 and p_5 at $\mathcal{R}$, with upload capacities of $(0.7, 0.8, 1.0)$ (in Mbps) at p_1, p_2, p_3, respectively, and available link capacities of $(0.4, 0.5, 0.5, 0.8, 0.6, 0.9)$ (in Mbps) on the six links.

The infeasible case: The overlay topology is unable to support p_3, p_4 and p_5 at $\mathcal{R}$, with upload capacities of $(0.7, 0.8, 0.6)$ (in Mbps) at p_1, p_2 and p_3. Available link capacities are the same as in the feasible case.

The step length sequences used in our experiments are $\alpha_n = 1/n$ and $\beta_n = 1/(10\sqrt{n})$. Results for the two experiments are illustrated in Fig. 4.3 and Fig. 4.4, respectively.

Fig. 4.3(a) depicts the convergence of allocated link rates, x_{ij}'s, in the feasible case, which are all initiated to 400 Kbps. In the first iteration, p_3 and p_4 find that there is insufficient bandwidth on link $(1, 3)$ and $(1, 4)$, respectively, as p_1's aggregate sending rate exceeds its upload capacity. Based on the feedbacks from its downstream peers, p_1

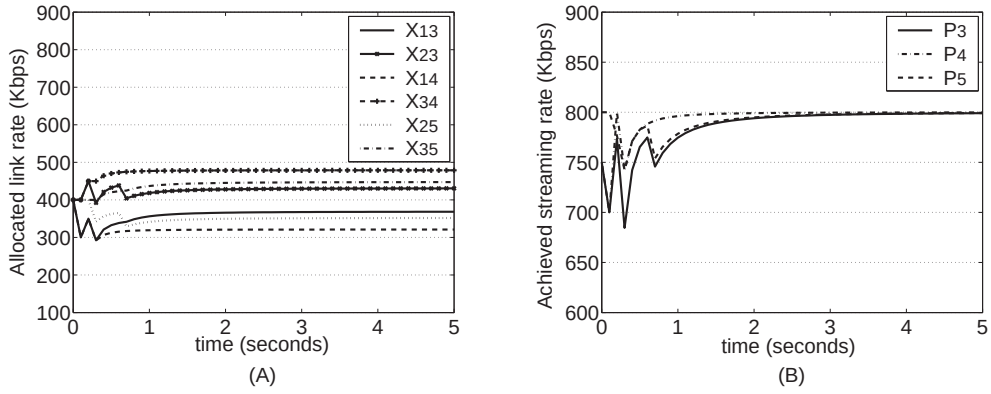


Figure 4.3: Convergence in the example topology: the feasible case.

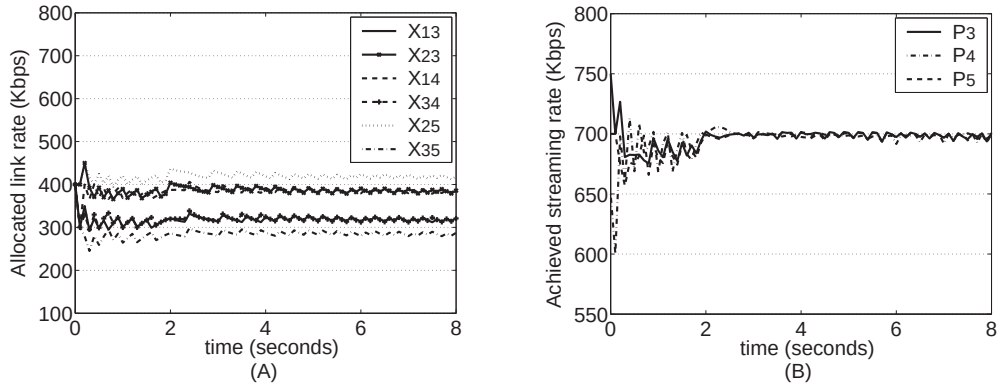


Figure 4.4: Convergence in the example topology: the infeasible case.

decreases x_{13} and x_{14}. In the next iteration, as the aggregate download link rates at p_3 and p_4 fall below $\mathcal{R}$, they increase the allocated rates on their download links. During the convergence, x_{34} and x_{35} keep increasing as p_3's spare upload capacity is being utilized, and the rate vector quickly converges to a feasible solution, $(370, 430, 322, 478, 352, 448)$. Correspondingly, Fig. 4.3(b) shows the convergence of the actually achieved streaming rates at the peers during the bandwidth allocation process. Though there is a temporary decrease initially, these rates steadily increase to reach the required rate.

Fig. 4.4 illustrates the convergence in the infeasible case, in which the maximum throughput achieved at p_3, p_4 and p_5 is 700 Kbps. We observe that while their initial

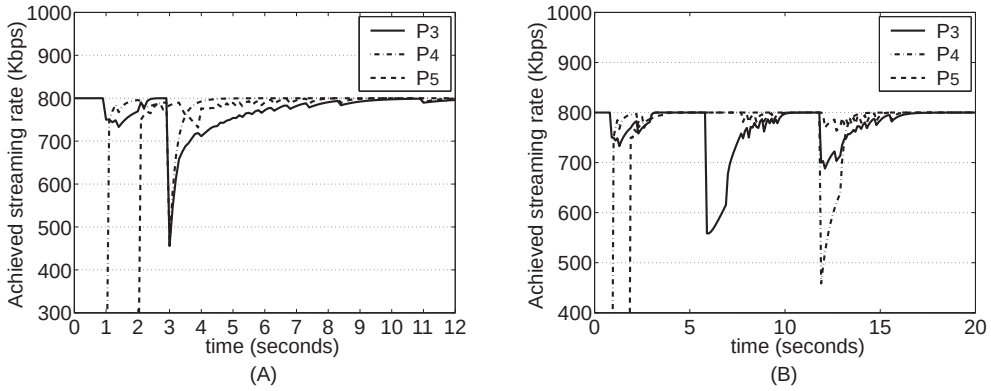


Figure 4.5: Convergence in the example topology: the dynamic case.

streaming rates are $750, 650, 700$ Kbps, respectively, the rates quickly converge to 700 Kbps at all three peers. This reveals that our implementation is able to achieve fair allocation among the peers, when there is insufficient "supply" of bandwidth in the overlay. The observations we have had in both feasible and infeasible cases have supported Corollary 1 in Sec. 4.2.

Experiment B (effects of dynamics): We next investigate how our implementation converges in dynamic environments, again with the example topology in Fig. 4.2. In this experiment, upload capacities at p_1, p_2 and p_3 are $(0.7, 1.0, 1.5)$ (in Mbps) and available link capacities on the six links are $(0.4, 0.8, 0.5, 0.8, 0.35, 0.9)$ (in Mbps). We study two cases, and show our results in Fig. 4.5(a) and (b), respectively.

Dynamics occur during convergence: p_1 and p_2 already exist in the overlay session. p_3 joins the session at time 0, p_4 joins at 1 second, p_5 joins at 2 seconds, and then p_1 leaves the network at 3 seconds.

Dynamics occur after convergence: p_1 and p_2 already exist in the overlay session. p_3 joins the session at time 0, p_4 joins at 1 seconds, and p_5 joins at 2 seconds. The available bandwidth on link $(1, 3)$ is then decreased to 0.1 Mbps at 6 seconds. Finally, p_1 leaves

the session at 12 seconds.

In the first case, comparing its results in Fig. 4.5(a) with Fig. 4.3(b), we can see that the convergence process is slightly prolonged due to peer dynamics, but is still performed rapidly. In the second case, after all three peers have joined the session and their rates stabilize, at 6 seconds, the decrease of available bandwidth on link $(1, 3)$ causes the decrease of p_3's streaming rate. As p_3 attempts to acquire additional bandwidth allocations from p_2, it further affects p_5. p_3 and p_5 then further adjust their bandwidth allocation, while p_4 is not affected. After the rates converge again, at 12 seconds, p_1's departure causes all three peers to cooperate in another round of rate adjustment, which quickly converges to a new feasible bandwidth allocation.

Experiment C (large networks): We are now ready to investigate how the asynchronous implementation of our iterative algorithm converges in more realistic and larger networks, generated with the BRITE topology generator [64]. In this set of experiments, peers are classified into two classes: 30% of them are Ethernet peers, with 10 Mbps upload capacities; the remainder are ADSL peers, with $0.4 - 0.6$ Mbps upload capacities. The streaming server is an Ethernet host. The message passing delays on overlay links are sampled from the distribution of pairwise ping times between PlanetLab nodes [1]. Available link bandwidths are chosen from the distribution of measured capacities between PlanetLab nodes as well [4]. As our bandwidth allocation algorithm is orthogonal to peer selection strategies used by the streaming applications, we apply random peer selection as our topology construction protocol. The experiment is further divided into two parts.

Exp. C. 1: To investigate the scalability of the protocol, we examine its convergence speed in networks of different sizes and various numbers of upstream peers for each peer

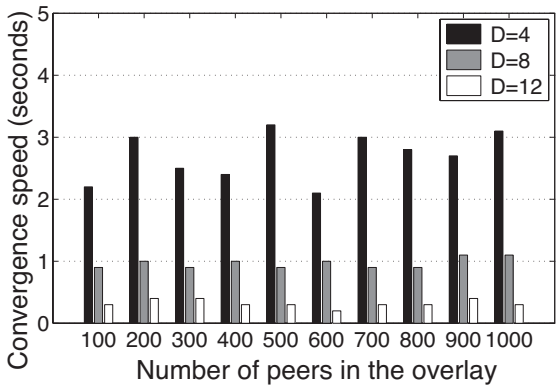


Figure 4.6: Converge speed in large random networks.

(D) in a static setting, without peer dynamics.

The results in Fig. 4.6 show that our algorithm scales very well with the increase of network sizes. This reveals that, in realistic large networks, by adjusting its bandwidth allocation with peers in its neighborhood, each peer can quickly obtain the required streaming bandwidth. The convergence is faster when a peer has more upstream peers.

Exp. C. 2: We then simulate a practical dynamic streaming session with $\mathcal{R} = 800$ Kbps, and monitor the achieved streaming rates at the peers during a 10-minute period. In the session, 200 peers join and depart following an On/Off model, with On/Off periods both following an exponential distribution with an expected length of T seconds. Each peer executes the following: Upon joining, it randomly selects D upstream peers and executes the bandwidth allocation algorithm. During such execution, if its achieved streaming rate is below $\mathcal{R}$ for 2 seconds, it randomly adds a new upstream peer if it currently has fewer than D upstream peers (due to peer failures), or randomly switches to new upstream peers otherwise.

The results in Fig. 4.7 demonstrate that our algorithm can provide the peers with steady streaming rates under high peer churn rates. In the case that each peer joins/leaves

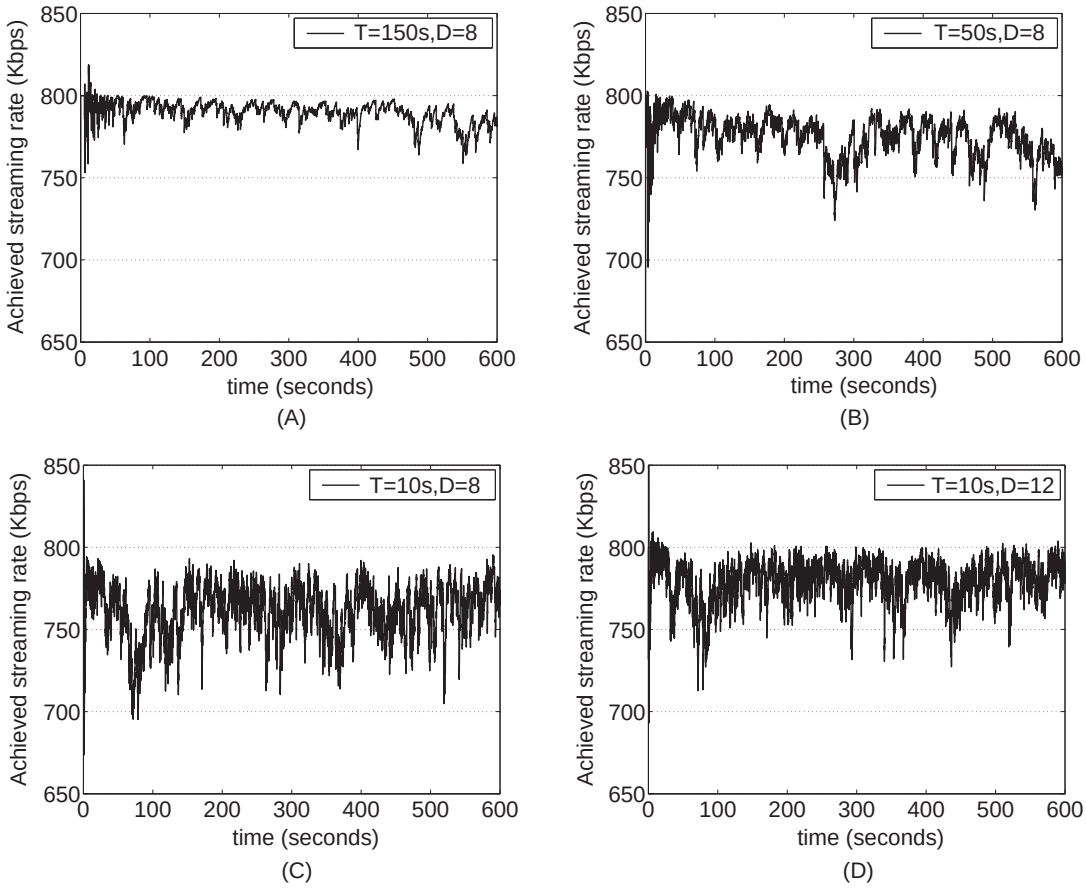


Figure 4.7: Average achieved streaming rate in a dynamic streaming session with 200 peers.

every 10 seconds, with 200 peers, every second there are 20 peer joins/departures on average. Even in such an extremely dynamic environment, the streaming rates at existing peers are rather satisfactory any time during the streaming session. Comparing Fig. 4.7(c) with (d), we can see that the streaming rates are better if each peer has more upstream peers.

We have also experimented with varying available overlay link bandwidths. However, as the available link capacities sampled from those between PlanetLab nodes are generally much larger than $\mathcal{R}$, such variations do not materially affect our results. We choose not

to present further details of these experiments.

4.5 Summary

In this chapter, we propose a practical algorithm that allocates the limited "supply" of bandwidth in a P2P streaming session, so that the "demand" of streaming playback rate can be satisfied on all the peers. We model the problem of streaming rate satisfaction as a feasibility problem, defined as a set of linear constraints. We further propose an iterative algorithm, which converges to a feasible solution if it exists, adapts well to dynamics, and can be implemented in a fully decentralized fashion, in both the synchronous and asynchronous cases. As salient advantages of our algorithm, it does not rely on *a priori* knowledge of available upload and link bandwidth, and is able to maximally guarantee the required streaming rate at each peer even in case of persistent peer dynamics and network changes.

4.6 Proofs

4.6.1 Proof of Theorem 1

Let $\tilde{\lambda}_{ij}^{(n)} = \lambda_j^{(n)}, \ \forall \ (i,j) \in \mathcal{A}$. The iterative updates in (4.4) can be expressed again in the following form:

$$x^{(n+1)} = x^{(n)} + \alpha_n \tilde{\lambda}^{(n)} - \beta_n e^{(n)}$$

where $\tilde{\lambda}^{(n)} = (\tilde{\lambda}_{ij}^{(n)}, \ \forall \ (i,j) \in \mathcal{A})$, and $e^{(n)} = (e_{ij}^{(n)}, \ \forall \ (i,j) \in \mathcal{A})$. We first prove a lemma:

Lemma 1. *If $X_R \cap X_C \neq \phi$, let $\tilde{x}$ be a feasible solution to **LC**, the following holds for*

all sufficiently large n for which $x^{(n)} \notin X_R \cap X_C$*:*

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - r\alpha_n,$$

where $r > 0$.

Proof: Choose $\tilde{r} > 0$ such that for all x satisfying $\|x - \tilde{x}\| \leq \tilde{r}$, $x \in X_R \cap X_C$. For every $x \in X_R \cap X_C$, we have

$$(\tilde{\lambda}^{(n)}, x^{(n)} - x) = \sum_{(i,j)\in\mathcal{A}} \tilde{\lambda}_{ij}^{(n)}(x_{ij}^{(n)} - x_{ij}) = \sum_{j\in\mathcal{N}} \lambda_j^{(n)}(\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^{(n)} - \sum_{i:(i,j)\in\mathcal{A}} x_{ij}).$$

For all $j \in \mathcal{N}$ where $\lambda_j^{(n)} > 0$, $\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^{(n)} < R$, and for all $x \in X_R$, $\sum_{i:(i,j)\in\mathcal{A}} x_{ij} \geq R$. We derive

$$(\tilde{\lambda}^{(n)}, x^{(n)} - x) \leq 0.$$

Let $x = \tilde{x} - \tilde{r}\frac{\tilde{\lambda}^{(n)}}{\|\tilde{\lambda}^{(n)}\|}$. Then $x \in X_R \cap X_C$. We have

$$(\tilde{\lambda}^{(n)}, x^{(n)} - x) = (\tilde{\lambda}^{(n)}, x^{(n)} - \tilde{x} + \tilde{r}\frac{\tilde{\lambda}^{(n)}}{\|\tilde{\lambda}^{(n)}\|}) \leq 0.$$

Therefore, $(\tilde{\lambda}^{(n)}, x^{(n)} - \tilde{x}) \leq -\tilde{r}\|\tilde{\lambda}^{(n)}\|$.

Next, for every $x \in X_R \cap X_C$, we have the following:

$$(e^{(n)}, x^{(n)} - x) = \sum_{(i,j)\in\mathcal{A}} e_{ij}^{(n)} x_{ij}^{(n)} - \sum_{(i,j)\in\mathcal{A}} e_{ij}^{(n)} x_{ij}.$$

The link rates $x_{ij}^{(n)}$'s for which $e_{ij}^{(n)} > 0$ can be categorized into two groups: one contains those which violate an upload capacity constraint in (4.2), and the other contains those

which do not violate (4.2), but violate an available link capacity constraint in (4.3). Let $N^{(n)}$ be the set of peers whose upload capacities are exceeded. Let $A^{(n)}$ be the set of links (i,j)'s, whose available bandwidths are exceeded, but $i \neq N^{(n)}$. Thus

$$(e^{(n)}, x^{(n)} - x) = \sum_{i \in N^{(n)}} (\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(n)} - \sum_{j:(i,j)\in\mathcal{A}} x_{ij}) + \sum_{(i,j)\in A^{(n)}} (x_{ij}^{(n)} - x_{ij}).$$

For all $i \in N^{(n)}$, $\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(n)} > \mathcal{O}_i$. For all $(i,j) \in A^{(n)}$, $x_{ij}^{(n)} > c_{ij}$. For all $x \in X_C$, $\sum_{j:(i,j)\in\mathcal{A}} x_{ij} \leq O_i$, and $x_{ij} \leq c_{ij}$. We then derive

$$(e^{(n)}, x^{(n)} - x) \geq 0.$$

Let $x = \tilde{x} + \tilde{r}\frac{e^{(n)}}{\|e^{(n)}\|}$. Then $x \in X_R \cap X_C$. We have

$$(e^{(n)}, x^{(n)} - x) = (e^{(n)}, x^{(n)} - \tilde{x} - \tilde{r}\frac{e^{(n)}}{\|e^{(n)}\|}) \geq 0.$$

Therefore,

$$-(e^{(n)}, x^{(n)} - \tilde{x}) \leq -\tilde{r}\|e^{(n)}\|. \tag{4.7}$$

We divide our discussions into three cases:

(1) $x^{(n)} \notin X_R$ and $x^{(n)} \in X_C$

In this case, $\|\tilde{\lambda}^{(n)}\| > 0$, $e^{(n)} = 0$. By definition of $\tilde{\lambda}^{(n)}$, we know $\|\tilde{\lambda}^{(n)}\|$ is bounded, *i.e.*, $0 < \tilde{a} \leq \|\tilde{\lambda}^{(n)}\| \leq \tilde{A}$. Then

$$\begin{aligned} \|x^{(n+1)} - \tilde{x}\|^2 &= \|x^{(n)} + \alpha_n\tilde{\lambda}^{(n)} - \tilde{x}\|^2 = \|x^{(n)} - \tilde{x}\|^2 + \alpha_n^2\|\tilde{\lambda}^{(n)}\|^2 + 2\alpha_n(\tilde{\lambda}^{(n)}, x^{(n)} - \tilde{x}) \\ &\leq \|x^{(n)} - \tilde{x}\|^2 + \tilde{A}^2\alpha_n^2 - 2\tilde{a}\tilde{r}\alpha_n. \end{aligned}$$

As $\alpha_n \to 0$ for sufficiently large n, we have $\alpha_n \leq \frac{\tilde{a}\tilde{r}}{\tilde{A}^2}$. Thus

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - \tilde{a}\tilde{r}\alpha_n.$$

(2) $x^{(n)} \in X_R$ and $x^{(n)} \notin X_C$

In this case, $\tilde{\lambda}^{(n)} = 0$ and $\|e^{(n)}\| > 0$. Let $L = |\mathcal{A}|$. We know $1 \leq \|e^{(n)}\|^2 \leq L$. Then

$$\begin{aligned} \|x^{(n+1)} - \tilde{x}\|^2 &= \|x^{(n)} - \beta_n e^{(n)} - \tilde{x}\|^2 = \|x^{(n)} - \tilde{x}\|^2 + \beta_n^2 \|e^{(n)}\|^2 - 2\beta_n (e^{(n)}, x^{(n)} - \tilde{x}) \\ &\leq \|x^{(n)} - \tilde{x}\|^2 + L\beta_n^2 - 2\tilde{r}\beta_n. \end{aligned}$$

As $\beta_n \to 0$, $\frac{\alpha_n}{\beta_n} \to 0$, we have $\beta_n \leq \frac{\tilde{r}}{L}$ and $\beta_n > \alpha_n$. Thus

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - \tilde{r}\beta_n \leq \|x^{(n)} - \tilde{x}\|^2 - \tilde{r}\alpha_n.$$

(3) $x^{(n)} \notin X_R$ and $x^{(n)} \notin X_C$

In this case, $\|\tilde{\lambda}^{(n)}\| > 0$ and $\|e^{(n)}\| > 0$. We have

$$\begin{aligned} & \|x^{(n+1)} - \tilde{x}\|^2 = \|x^{(n)} + \alpha_n \tilde{\lambda}^{(n)} - \beta_n e^{(n)} - \tilde{x}\|^2 \\ = \; & \|x^{(n)} - \tilde{x}\|^2 + \alpha_n^2 \|\tilde{\lambda}^{(n)}\|^2 + 2\alpha_n (\tilde{\lambda}^{(n)}, x^{(n)} - \tilde{x}) + \beta_n^2 \|e^{(n)}\|^2 - 2\beta_n (e^{(n)}, x^{(n)} - \tilde{x}) \\ & -2\alpha_n \beta_n (\tilde{\lambda}^{(n)}, e^{(n)}) \\ \leq \; & \|x^{(n)} - \tilde{x}\|^2 + \tilde{A}^2 \alpha_n^2 - 2\tilde{a}\tilde{r}\alpha_n + L\beta_n^2 - 2\tilde{r}\beta_n \leq \|x^{(n)} - \tilde{x}\|^2 - \tilde{a}\tilde{r}\alpha_n - \tilde{r}\alpha_n. \end{aligned}$$

Taking $r = \min(\tilde{a}\tilde{r}, \tilde{r})$, we have for all three cases:

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - r\alpha_n.$$

Proof of Theorem 1: We show that there exists a sufficiently large $\tilde{n}$, such that $x^{(\tilde{n})} \in X_R \cap X_C$. We prove by contradiction.

Assume that there exists a n' such that $x^{(n)} \neq X_R \cap X_C$ for all $n \geq n'$. From Lemma 1, we know $\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - r\alpha_n$. Then we sum up all such inequalities for $n = n'$ to $n = n' + m$ and obtain

$$\|x^{(n'+m+1)} - \tilde{x}\|^2 \leq \|x^{(n')} - \tilde{x}\|^2 - r \sum_{n=n'}^{n'+m} \alpha_n,$$

which implies that $\|x^{(n'+m+1)} - \tilde{x}\| \to -\infty$ when $m \to \infty$, since $\sum_{n=n'}^{n'+m} \alpha_n \to \infty$. This is impossible as $\|x^{(n'+m+1)} - \tilde{x}\| \geq 0$. Therefore, there always exists one sufficiently large $\tilde{n}$ such that $x^{(\tilde{n})} \in X_R \cap X_C$. Based on the algorithm in (4.4), we know $x^{(n)} = x^{(\tilde{n})},\ \forall\ n > \tilde{n}$. Thus Theorem 1 is proven. □

4.6.2 Proof of Theorem 2

Let X'_R be the region defined by (4.6), *i.e.*, $X'_R = \{x : \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \geq \mathcal{R}_{\max},\ \forall\ j \in \mathcal{N}\}$. The feasible region of problem **LC'** can be represented as $X' = X'_R \cap X_C$. We first prove two lemmas:

Lemma 2. *Let $\tilde{x}$ be a feasible solution to **LC'**. The following results hold for sufficiently large n,* where $r_1 > 0, r_2 > 0$:

(1) If $x^{(n)} \notin X_C$, $\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - r_1\beta_n$;

(2) If $x^{(n)} \notin X'_R$ and $x^{(n)} \in X_C$, $\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 + r_2\alpha_n$.

Proof: For every $\tilde{x} \in X'_R \cap X_C$, we have the following:

$$(\tilde{\lambda}^{(n)}, x^{(n)} - \tilde{x}) = \sum_{j\in\mathcal{N}} \lambda_j^{(n)} \Big(\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^{(n)} - \sum_{i:(i,j)\in\mathcal{A}} \tilde{x}_{ij} \Big)$$

For all j where $\lambda_j^{(n)} > 0$, $\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^{(n)} < R$. For $\tilde{x} \in X_R'$, $\sum_{i:(i,j)\in\mathcal{A}} x_{ij} \geq R_{\max}$. Let N be the total number of peers in the network, *i.e.*, $N = |\mathcal{N}|$. We have

$$(\tilde{\lambda}^{(n)}, x^{(n)} - x) \leq NR(R - R_{\max})$$

Next, choose $\tilde{r} > 0$ such that for all x satisfying $\|x - \tilde{x}\| \leq \tilde{r}$, $x \in X_R' \cap X_C$. With the same proof as that for inequality (4.7) in Lemma 1, for every $\tilde{x} \in X_R' \cap X_C$, we have

$-(e^{(n)}, x^{(n)} - \tilde{x}) \leq -\tilde{r}\|e^{(n)}\|$.

We next prove the two results in Lemma 2:

(1) If $x^{(n)} \notin X_C$, we can further divide the discussions into two sub cases:

(a) $x^{(n)} \notin X_C$ and $x^{(n)} \in X_R$

In this case, $\tilde{\lambda}^{(n)} = 0$ and $\|e^{(n)}\| > 0$. Let L be the total number of links in the network, *i.e.*, $L = |\mathcal{A}|$. We know $1 \leq \|e^{(n)}\|^2 \leq L$. Then

$$\begin{aligned} \|x^{(n+1)} - \tilde{x}\|^2 &= \|x^{(n)} - \beta_n e^{(n)} - \tilde{x}\|^2 = \|x^{(n)} - \tilde{x}\|^2 + \beta_n^2\|e^{(n)}\|^2 - 2\beta_n(e^{(n)}, x^{(n)} - \tilde{x}) \\ &\leq \|x^{(n)} - \tilde{x}\|^2 + L\beta_n^2 - 2\tilde{r}\beta_n \end{aligned}$$

As $\beta_n \to 0$, for sufficiently large n, we have $\beta_n \leq \frac{\tilde{r}}{L}$. Therefore,

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - \tilde{r}\beta_n$$

(b) $x^{(n)} \notin X_C$ and $x^{(n)} \notin X_R$.

In this case, $\|\tilde{\lambda}^{(n)}\| > 0$ and $\|e^{(n)}\| > 0$. From the definition of $\tilde{\lambda}^{(n)}$, we know $\|\tilde{\lambda}^{(n)}\|$

is bounded, *i.e.*, $0 < \tilde{a} \leq \|\tilde{\lambda}^{(n)}\| \leq \tilde{A}$. Then

$$\begin{aligned} & \|x^{(n+1)} - \tilde{x}\|^2 = \|x^{(n)} + \alpha_n \tilde{\lambda}^{(n)} - \beta_n e^{(n)} - \tilde{x}\|^2 \\ = \; & \|x^{(n)} - \tilde{x}\|^2 + \alpha_n^2 \|\tilde{\lambda}^{(n)}\|^2 + 2\alpha_n (\tilde{\lambda}^{(n)}, x^{(n)} - \tilde{x}) + \beta_n^2 \|e^{(n)}\|^2 - 2\beta_n (e^{(n)}, x^{(n)} - \tilde{x}) \\ & -2\alpha_n \beta_n (\tilde{\lambda}^{(n)}, e^{(n)}) \\ \leq \; & \|x^{(n)} - \tilde{x}\|^2 + \tilde{A}^2 \alpha_n^2 + 2NR(R - R_{\max})\alpha_n + L\beta_n^2 \; - 2\tilde{r}\beta_n \end{aligned}$$

As $\alpha_n \to 0$ and $\beta_n \to 0$ for sufficiently large n, we have $\alpha_n \leq \frac{NR(R-R_{\max})}{\tilde{A}^2}$ and $\beta_n < \frac{\tilde{r}}{L}$. Thus

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 + 3NR(R - R_{\max})\alpha_n - \tilde{r}\beta_n$$

As $\frac{\alpha_n}{\beta_n} \to 0$, we get $\alpha_n \leq \frac{\tilde{r}\beta_n}{6NR(R-R_{\max})}$. Thus

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - \frac{\tilde{r}}{2}\beta_n$$

Let $r_1 = \frac{\tilde{r}}{2}$, we have for the above two sub cases:

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - r_1\beta_n$$

(2) If $x^{(n)} \in X_C$ and $x^{(n)} \notin X'_R$

In this case, $x^{(n)} \notin X_R$ as well. Thus $\|\tilde{\lambda}^{(n)}\| > 0$ and $e^{(n)} = 0$. We get

$$\begin{aligned} & \|x^{(n+1)} - \tilde{x}\|^2 = \|x^{(n)} + \alpha_n \tilde{\lambda}^{(n)} - \tilde{x}\|^2 = \|x^{(n)} - \tilde{x}\|^2 + \alpha_n^2 \|\tilde{\lambda}^{(n)}\|^2 + 2\alpha_n (\tilde{\lambda}^{(n)}, x^{(n)} - \tilde{x}) \\ \leq \; & \|x^{(n)} - \tilde{x}\|^2 + \tilde{A}^2 \alpha_n^2 + 2NR(R - R_{\max})\alpha_n \end{aligned}$$

As $\alpha_n \to 0$, we have $\alpha_n \leq \frac{NR(R-R_{\max})}{\tilde{A}^2}$. Thus

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 + 3NR(R - R_{\max})\alpha_n$$

Let $r_2 = 3NR(R - R_{\max})$, we obtain

$$\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 + r_2\alpha_n.$$

□

Lemma 3. *For any sufficiently large n, there exists an infinite sequence $n < n_1 < n_2 < n_3 < \ldots$, such that $x^{(n_i)} \in X'_R \cap X_C$, for all $i = 1, 2, 3, \ldots$.*

Proof: We prove by contradiction. Assume there exists a n' such that $x^{(n)} \notin X'_R \cap X_C$ for all $n > n'$.

Without loss of generality, we assume the following: (1) $x^{(n'+1)} \in X_C$ but $x^{(n'+1)} \notin X'_R$; (2) $x^{(n'+m)} \notin X_C$; (3) For all $n > n'$, if $x^{(n)} \notin X_C$, $x^{(n+1)} \in X_C$. From Lemma 2, we know that (1) If $x^{(n)} \notin X_C$, $\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 - r_1\beta_n$; (2) If $x^{(n)} \notin X'_R$ and $x^{(n)} \in X_C$, $\|x^{(n+1)} - \tilde{x}\|^2 \leq \|x^{(n)} - \tilde{x}\|^2 + r_2\alpha_n$. We sum up all the inequalities for $n = n' + 1$ to $n = n' + m$, and obtain

$$\|x^{(n'+m+1)} - \tilde{x}\|^2 \leq \|x^{(n'+1)} - \tilde{x}\|^2 + Z$$

where

$$Z = r_2 \sum_{n=n'+1}^{n'+p_1-1} \alpha_n - r_1\beta_{n'+p_1} + r_2 \sum_{n=n'+p_1+1}^{n'+p_1+p_2-1} \alpha_n - r_1\beta_{n'+p_1+p_2} + \ldots$$

$$+ \quad r_2 \sum_{n=n'+\sum_{l=1}^{i-1} p_l+1}^{n'+\sum_{l=1}^{i} p_l-1} \alpha_n - r_1\beta_{n'+\sum_{l=1}^{i} p_l} + \ldots + r_2 \sum_{n=n'+m-p_k+1}^{n'+m-1} \alpha_n - r_1\beta_{n'+m} \quad (4.8)$$

where $p_i < \infty$ is the number of steps for a $x^{(n)} \notin X_C$ to move to the next $x^{(n)} \notin X_C$, for all $i = 1, 2, \ldots, k$ and $m = \sum_{l=1}^{k} p_l$.

As $\alpha_n \to 0, \alpha_n < \beta_n$ for sufficiently large n, we have $\alpha_n < \frac{r_1\beta_{n'+\sum_{l=1}^{i} p_l}}{2r_2(p_i-1)}$, for all $n = n' + \sum_{l=1}^{i-1} p_l + 1, \ldots, n' + \sum_{l=1}^{i} p_l - 1, i = 1, \ldots, k$. Therefore

$$Z \leq -\frac{r_1}{2}(\beta_{n'+p_1} + \beta_{n'+p_1+p_2} + \ldots + \beta_{n'+\sum_{l=1}^{i} p_l} + \ldots + \beta_{n'+m}) = -\frac{r_1}{2}\sum_{i=1}^{k} \beta_{n'+\sum_{l=1}^{i} p_l}$$

Thus

$$\|x^{(n'+m+1)} - \tilde{x}\|^2 \leq \|x^{(n'+1)} - \tilde{x}\|^2 - \frac{r_1}{2}\sum_{i=1}^{k} \beta_{n'+\sum_{l=1}^{i} p_l}$$

If $m \to \infty$, we know $k \to \infty$ and $\sum_{i=1}^{k} \beta_{n'+\sum_{l=1}^{i} p_l} \to \infty$. Thus $\|x^{(n'+m+1)} - \tilde{x}\| \to -\infty$, which is impossible as $\|x^{(n'+m+1)} - \tilde{x}\| \geq 0$. Therefore, our assumption is not correct. Thus the lemma is proven. □

Proof of Theorem 2: We show $\rho(x^{(n)}, X') \leq \delta$, for any δ. Based on Lemma 3, we know there is a sequence $n_1 < n_2 < n_3 < \ldots$, for which $x^{(n_i)} \in X'$, for all $i = 1, 2, \ldots$. We discuss three cases:

(1) $n = n_i$, for some $i \in \{1, 2, \ldots\}$. Thus $x^{(n)} \in X'$, and $\rho(x^{(n)}, X') = 0$.

(2) $n = n_i + 1$. In this case, $x^{(n)} = x^{(n_i)} + \alpha_{n_i}\tilde{\lambda}^{(n_i)}$. As $\alpha_{n_i} \to \infty$ for sufficiently large n_i, we have

$$\|x^{(n)} - x^{(n_i)}\| = \|x^{(n_i)} + \alpha_{n_i}\tilde{\lambda}^{(n_i)} - x^{(n_i)}\| = \alpha_{n_i}\|\tilde{\lambda}^{(n_i)}\| \leq \tilde{A}\alpha_{n_i} < \delta$$

Then, we have

$$\rho(x^{(n)}, X') \leq \rho(x^{(n_i)}, X') + \|x^{(n)} - x^{(n_i)}\| \leq \delta. \tag{4.9}$$

(3) $n_i + 1 < n < n_{i+1}$. From (4.9), we know there exists a $x^* \in X'$, such that $\|x^{(n_i+1)} - x^*\| \leq \delta$. As $x^{(n)} \notin X'$, from Lemma 2, we know that (A) If $x^{(n)} \notin X_C$, $\|x^{(n+1)} - x^*\|^2 \leq \|x^{(n)} - x^*\|^2 - r_1\beta_n$; (B) If $x^{(n)} \notin X'_R$ and $x^{(n)} \in X_C$, $\|x^{(n+1)} - x^*\|^2 \leq \|x^{(n)} - x^*\|^2 + r_2\alpha_n$. We sum up the inequalities for $n = n_i + 1$ to $n - 1$, and can obtain

$$\|x^{(n)} - x^*\|^2 \leq \|x^{(n_i+1)} - x^*\|^2 + Z$$

where $Z \leq 0$ is in the similar form as (4.8). Thus

$$\rho(x^{(n)}, X') \leq \|x^{(n)} - x^*\| \leq \delta$$

□

4.6.3 Proof of Theorem 3

We prove the case that there is a feasible solution to **LC** and omit the proof for the infeasible case, which can be done in the similar way as Theorem 2 is proven. We first show the following lemma:

Lemma 4. *At any time t, if there are any updates to any allocated rates $x_{ij}(t)$, $\forall\ (i,j) \in \mathcal{A}$, the rate vector $x(t)$ is improving towards $\tilde{x}$, which is a feasible solution of **LC**, i.e.,*

$$\|x(t+1) - \tilde{x}\|^2 \leq \|x(t) - \tilde{x}\|^2 - r\gamma(t),$$

where $r > 0$ and sequence $\gamma(t)$ satisfies $\lim_{t\to\infty}\gamma(t) = 0, \sum_{t=1}^{\infty}\gamma(t) = \infty$.

Proof: At time t, let $P1$ be the set of peers that are increasing the allocated rates on their download links. Let $P2$ be the set of peers that are decreasing the allocated rates on their upload links. We have

$$x_{ij}(t+1) = \begin{cases} x_{ij}(t) + \alpha_{n_j}\lambda_j(t) & \text{if } j \in P1, \\ x_{ij}(t) - \beta_{m_i}e_{ij}(t) & \text{if } i \in P2, \\ x_{ij}(t) & \text{otherwise.} \end{cases}$$

Then

$$\begin{aligned} \|x(t+1) - \tilde{x}\|^2 &= \sum_{(i,j)\in\mathcal{A}}(x_{ij}(t) - \tilde{x}_{ij})^2 + \sum_{(i,j):j\in P1}(\alpha_{n_j}^2\lambda_j^2(t) + 2\alpha_{n_j}\lambda_j(t)(x_{ij}(t) - \tilde{x}_{ij})) \\ &\quad + \sum_{(i,j):i\in P2}(\beta_{m_i}^2 e_{ij}^2(t) - 2\beta_{m_i}e_{ij}(t)(x_{ij}(t) - \tilde{x}_{ij})). \end{aligned}$$

Let $\Lambda = \sum_{(i,j):j\in P1}(\alpha_{n_j}^2\lambda_j^2(t) + 2\alpha_{n_j}\lambda_j(t)(x_{ij}(t) - \tilde{x}_{ij}))$,
$\Theta = \sum_{(i,j):i\in P2}(\beta_{m_i}^2 e_{ij}^2(t) - 2\beta_{m_i}e_{ij}(t)(x_{ij}(t) - \tilde{x}_{ij}))$. We have

$$\Lambda = \sum_{(i,j):j\in P1}\lambda_j^2(t)\alpha_{n_j}^2 + \sum_{j\in P1}2\lambda_j(t)\alpha_{n_j}(\sum_{i:(i,j)\in\mathcal{A}}x_{ij}(t) - \sum_{i:(i,j)\in\mathcal{A}}\tilde{x}_{ij}).$$

For all $j \in P1$, as $\lambda_j(t) > 0$, $\sum_{i:(i,j)\in\mathcal{A}}x_{ij}(t) < R$. For all $\tilde{x} \in X_R$, $\sum_{i:(i,j)\in\mathcal{A}}\tilde{x}_{ij} \geq R$. Let D be the maximum number of upstream peers each peer has. Then

$$\Lambda \leq \sum_{j\in P1}(DR^2\alpha_{n_j}^2 - 2\tilde{r}\alpha_{n_j}),$$

where $\tilde{r} > 0$.

As $\alpha_{n_j} \to 0$, we have $\alpha_{n_j} < \frac{\tilde{r}}{DR^2}$. Let $\alpha(t) = \min(\alpha_{n_j},\ \forall\ j \in P1)$.

$$\Lambda \leq \sum_{j \in P1} (-\tilde{r}\alpha_{n_j}) \leq -|P1|\tilde{r}\alpha(t).$$

Next, we have

$$\Theta = \sum_{(i,j):i \in P2} \beta_{m_i}^2 e_{ij}^2(t) - 2 \sum_{(i,j):i \in P2} (\beta_{m_i} e_{ij}(t)(x_{ij}(t) - \tilde{x}_{ij}))$$

The link rates $x_{ij}(t)$'s for which $i \in P2$ can be categorized into two groups: one contains those which violate an upload capacity constraint in (4.2), and the other contains those which do not violate (4.2), but violate an available link capacity constraint in (4.3). Let $N(t)$ be the set of peers whose upload capacity is exceeded at time t. Let $A(t)$ be the set of links (i,j)'s, whose available bandwidths are exceeded at time t, but $i \neq N(t)$. We have

$$\sum_{(i,j):i \in P2} (\beta_{m_i} e_{ij}(t)(x_{ij}(t) - \tilde{x}_{ij})) = \sum_{i \in N(t)} \beta_{m_i} (\sum_{j:(i,j) \in \mathcal{A}} x_{ij}(t) - \sum_{j:(i,j) \in \mathcal{A}} \tilde{x}_{ij}) + \sum_{(i,j) \in A(t)} \beta_{m_i} (x_{ij}(t) - \tilde{x}_{ij})$$

As for all $i \in N(t)$, $\sum_{j:(i,j) \in \mathcal{A}} x_{ij}(t) > \mathcal{O}_i$, and for all $\tilde{x} \in X_C$, $\sum_{j:(i,j) \in \mathcal{A}} \tilde{x}_{ij} \leq O_i$, we get $\sum_{j:(i,j) \in \mathcal{A}} x_{ij}(t) - \sum_{j:(i,j) \in \mathcal{A}} \tilde{x}_{ij} > 0$. Similarly, since for all $(i,j) \in A(t)$, $x_{ij}(t) > \mathcal{C}_{ij}$, and for all $\tilde{x} \in X_C$, $\tilde{x}_{ij} \leq \mathcal{C}_{ij}$, we have $x_{ij}(t) - \tilde{x}_{ij} > 0$. Therefore, we derive

$$\sum_{(i,j):i \in P2} (\beta_{m_i} e_{ij}(t)(x_{ij}(t) - \tilde{x}_{ij})) \geq \sum_{i \in P2} \tilde{r}' \beta_{m_i} \geq 0,$$

where $\tilde{r}' > 0$.

Then, let D' be the maximum number of downstream peer each peer has. We get

$$\Theta = \sum_{(i,j):i\in P2} \beta_{m_i}^2 e_{ij}^2(t) - 2\sum_{(i,j):i\in P2} (\beta_{m_i} e_{ij}(t)(x_{ij}(t) - \tilde{x}_{ij})) \leq \sum_{i\in P2} (D'\beta_{m_i}^2 - 2\tilde{r}'\beta_{m_i})$$

As $\beta_{m_i} \to 0$, we have $\beta_{m_i} < \frac{\tilde{r}'}{D'}$. Let $\beta(t) = \min(\beta_{m_i},\ \forall\ i \in P2)$. We get

$$\Theta \leq -|P2|\tilde{r}'\beta(t).$$

Therefore,

$$\|x(t+1) - \tilde{x}\|^2 \leq \|x(t) - \tilde{x}\|^2 - |P1|\tilde{r}\alpha(t) - |P2|\tilde{r}'\beta(t).$$

Let $\gamma(t) = \min(\alpha(t), \beta(t))$, $r = |P1|\tilde{r} + |P2|\tilde{r}'$. We have

$$\|x(t+1) - \tilde{x}\|^2 \leq \|x(t) - \tilde{x}\|^2 - r\gamma(t),$$

and $\gamma(t)$ satisfies $\lim_{t\to\infty} \gamma(t) = 0, \sum_{t=1}^{\infty} \gamma(t) = \infty$. □

Proof of Theorem 3: We show that there exists a sufficiently large $\tilde{t}$, such that $x(\tilde{t}) \in X_R \cap X_C$. We prove by contradiction.

Assume that there exists a t' such that $x(t) \neq X_R \cap X_C$ for all $t \geq t'$. From Lemma 4, we know $\|x(t+1) - \tilde{x}\|^2 \leq \|x(t) - \tilde{x}\|^2 - r\gamma(t)$. Then we sum up all such inequalities for $t = t'$ to $t = t' + m$ and obtain

$$\|x(t'+m+1) - \tilde{x}\|^2 \leq \|x(t') - \tilde{x}\|^2 - r\sum_{t=t'}^{t'+m} \gamma(t)$$

which implies that $\|x(t'+m+1) - \tilde{x}\| \to -\infty$ when $m \to \infty$, since $\sum_{t=t'}^{t'+m} \gamma(t) \to \infty$. This is impossible as $\|x(t'+m+1) - \tilde{x}\| \geq 0$. Therefore, our assumption is not correct,

and there always exists one sufficiently large $\tilde{t}$ such that $x(\tilde{t}) \in X_R \cap X_C$. Based on the protocol, we know $x(t) = x(\tilde{t}),\ \forall\ t > \tilde{t}$. Thus Theorem 3 is proven. □

Chapter 5

Dynamic Bandwidth Auction in Multi-Overlay P2P Streaming

In mesh-based P2P streaming applications, multiple coexisting streaming overlays, corresponding to different channels of television programming or live events, are the norm rather than the exception. Generated with a modern codec, each overlay distributes a live media stream with a specific streaming rate. As compared to the single session scenario, it is a more challenging task to meet the demands of bandwidth at all participating peers when coexisting streaming overlays are considered, sharing the available upload bandwidth supplies in the P2P network.

Consider a typical scenario where multiple peers from different overlays are in conflict with one another, competing for limited upload bandwidth at the same streaming server or upstream peer in the network. In this case, the allocation of such upload bandwidth needs to be meticulously mediated with appropriate strategies, such that the streaming rate requirement of each overlay is satisfied at all their participating peers. It would be best if, at the same time, fairness can be achieved across different overlays, and costs of

streaming (*e.g.,* latencies) can be minimized.

Most existing P2P streaming studies focus on a single overlay, neglecting the conflict scenarios among coexisting overlays with respect to available bandwidth. Our study in this chapter represents the *first* in the category that focuses on such conflicts, by designing simple, decentralized, and effective tactical strategies to resolve inherent bandwidth conflicts among coexisting streaming overlays. Based on the rich foundation offered by game theory, we characterize the bandwidth conflicts among multiple overlays with *dynamic auction games*. Such games evolve over time, and involve repeated *auctions* in which competing downstream peers from different overlays *bid* for upload bandwidth at the same upstream peer, and the upstream peer allocates its upload bandwidth based on the bids. We design effective strategies to carry out the distributed auctions, which result in efficient bandwidth partition in the network across overlays.

With extensive theoretical analysis, we show that our decentralized game-theoretic strategies not only converge to a Nash equilibrium, but also lead to favorable outcomes in realistic asynchronous environments: We are able to obtain an optimal topology for each of the coexisting streaming overlays, in the sense that streaming rates are satisfied, and streaming costs are minimized. These topologies of coexisting overlays evolve and adapt to peer dynamics, fairly share peer upload bandwidth, and can be prioritized.

5.1 Multi-Overlay Streaming Model

5.1.1 Network model and assumptions

We consider a P2P live streaming network including multiple coexisting streaming overlays, each consisting of streaming servers and participating peers. Each server may serve more than one overlay, while each peer may also participate in multiple overlays. A

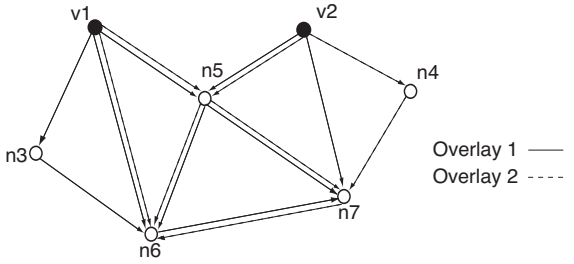


Figure 5.1: Two concurrent P2P streaming overlays: an example.

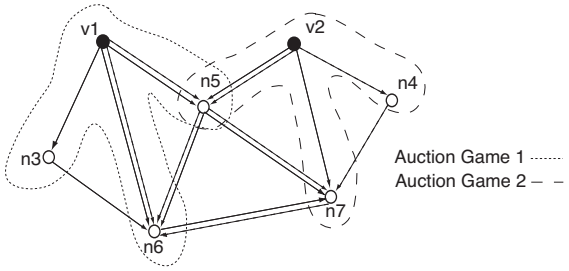


Figure 5.2: Decentralized auction games in the example overlays.

practical scenario of this case is that users behind a same gateway may watch different channels, while they appear as the same identity (peer) in the Internet with one same external IP address. Fig. 5.1 shows an example of two coexisting streaming overlays, each with two streaming servers and four participating peers.

In each streaming overlay, participating servers and peers form a mesh topology, in which any peer is served by its upstream peers (servers can be deemed as special upstream peers), and may serve one or more downstream peers at the same time. With respect to the construction of each overlay, as in the previous chapters, we assume there exists a standalone neighbor list maintenance mechanism, which bootstraps each new peer with a number of existing peers in an overlay, and maintains the number of neighbors for each peer during streaming as well. The application-layer links between peers in each overlay are established based on their media content availability during streaming.

Let $\mathcal{S}$ denote the set of all coexisting streaming overlays in the network. The topology of each overlay $s \in \mathcal{S}$ can be modeled as a directed graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{N}_s, \mathcal{A}_s)$, where $\mathcal{V}_s$ is the set of servers serving overlay s, $\mathcal{N}_s$ represents the set of participating peers, and $\mathcal{A}_s$ denotes the set of application-layer links in overlay s. Let R_s be the required streaming rate of the media stream distributed in overlay s. Let $\mathcal{V}$ be the set of all streaming servers in the network, *i.e.*, $\mathcal{V} = \cup_{s \in \mathcal{S}} \mathcal{V}_s$, and $\mathcal{N}$ be the set of all existing peers, *i.e.*,

$\mathcal{N} = \cup_{s \in \mathcal{S}} \mathcal{N}_s$. Let U_i denote the upload bandwidth at peer i, $\forall\ i \in \mathcal{V} \cup \mathcal{N}$.

Similar to our capacity assumption in Chapter 3, we assume that the last-mile upload bandwidth on each peer (including servers) constitutes the "supply" of bandwidth in the overlays, *i.e.*, bandwidth bottlenecks lie at the peers rather than at the core of the overlays. In addition, we assume that the download bandwidth of each peer is sufficient to support the required streaming rate(s) of the overlay(s) it participates in.

5.1.2 Auction game model

To resolve the bandwidth conflict on its upload link, each upstream peer i in the network, $\forall\ i \in \mathcal{V} \cup \mathcal{N}$, organizes a *dynamic bandwidth auction game*, referred to as auction i. In auction i, the "goods" for sale is the upload bandwidth of peer i with a total quantity of U_i, and the players are all the downstream peers of peer i in all overlays it participates in. Let j^s represent peer j in overlay s. The set of players in auction i can be expressed as $\{j^s,\ \forall\ j : (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}\}$. We note that in case a downstream peer j participates in multiple overlays, it is viewed as multiple players, each for one overlay. As each peer in an overlay may stream from multiple upstream peers in the overlay, a player j^s may concurrently bid for upload bandwidth in multiple auction games, each hosted by one upstream peer.

The auction games at the peers are dynamically carried out in a repeated fashion to resolve bandwidth conflicts over time. In each *bidding round* of auction i, each player submits its bid to peer i, declaring its requested share of upload bandwidth, as well as the unit price it is willing to pay. The upstream peer i then allocates shares of its upload capacity, U_i, to the players based on their bids. Let x_{ij}^s denote the upload bandwidth that player j^s requests from peer i, and p_{ij}^s denote the unit price it is willing to pay to

peer i. The bid from player j^s in auction i can be represented as a 2-tuple $b_{ij}^s = (p_{ij}^s, x_{ij}^s)$.

Such a distributed game model can be illustrated with the example in Fig. 5.2. In the example, there exist 7 auction games, two of which are marked: auction 1 at v_1 with 5 players 3^1 (peer $n3$ in overlay 1), 5^1, 5^2, 6^1 and 6^2, auction 2 at v_2, with 4 players 4^2, 5^1, 5^2 and 7^1, respectively.

5.2 Auction Strategies

Based on the above model, we have designed effective auction strategies to resolve bandwidth conflicts in multi-overlay streaming, including the allocation strategy taken by an upstream peer and the bidding strategy by downstream peers.

5.2.1 Allocation strategy

In auction i, the seller, upstream peer i, aims to maximize its revenue by selling its upload bandwidth U_i at the best prices. Given bids $b_{ij}^s = (p_{ij}^s, x_{ij}^s)$'s from all the players j^s ($\forall\ j : (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$), upstream peer i's allocation strategy can be represented by the following revenue maximization problem. Here, a_{ij}^s ($\forall\ j : (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$) is the bandwidth share to be allocated to each downstream peer j in each competing overlay s.

Allocation i:

$$\max \sum_{s \in \mathcal{S}} \sum_{j:(i,j) \in \mathcal{A}_s} p_{ij}^s a_{ij}^s \tag{5.1}$$

subject to

$$\sum_{s\in\mathcal{S}}\sum_{j:(i,j)\in\mathcal{A}_s} a_{ij}^s \le U_i,$$
$$0 \le a_{ij}^s \le x_{ij}^s, \qquad \forall\ j:(i,j)\in\mathcal{A}_s,\ \forall\ s\in\mathcal{S}.$$

Such an allocation strategy can be achieved in the following fashion:

Upstream peer i selects the highest bid price, e.g., *p_{ij}^s from player j^s, and allocates bandwidth $a_{ij}^s = \min(U_i, x_{ij}^s)$ to it. Then if it still has remaining bandwidth, it selects the second highest bid price and assigns the requested bandwidth to the corresponding player. This process repeats until peer i has allocated all its upload capacity, or bandwidth requests from all the players have been satisfied.* □

The above allocation strategy can be formally stated in the following formula:

$$a_{ij}^s = \min(x_{ij}^s, U_i - \sum_{p_{ik}^{s'}\ge p_{ij}^s, k^{s'}\neq j^s} a_{ik}^{s'}),\ \forall\ j:(i,j)\in\mathcal{A}_s,\ \forall\ s\in\mathcal{S}. \tag{5.2}$$

5.2.2 Bidding strategy

In each overlay $s \in \mathcal{S}$, a peer j may place its bids to multiple upstream peers, that can supply available media blocks to it. As a common objective, it wishes to achieve the required streaming rate for the overlay, and experiences minimum costs. We consider two parts of costs when peer j streams from peer i in overlay s: streaming cost — denoted by streaming cost function $D_{ij}^s(x_{ij}^s)$ — represents the streaming latency actually experienced by j; bidding cost — calculated by $p_{ij}^s x_{ij}^s$ — represents the bid peer j submits to peer i in overlay s. The bidding cost reflects the degree of *competition* and *demand* for bandwidth in the auctions at upstream peers. The overall cost at player j^s is the sum of the two parts from all its upstream peers, $\forall\ i:(i,j)\in\mathcal{A}_s$.

In this way, the preference for player j^s in deciding its bids in the auctions can be

expressed by the following cost minimization problem. Practically, each cost function D_{ij}^s should be non-decreasing and its value increases more rapidly when the requested bandwidth x_{ij}^s is larger (*i.e.* the property of convexity [23]). Therefore, without loss of generality, we assume the cost functions are non-decreasing, twice differentiable and strictly convex.

Bidding $\mathbf{j^s}$:

$$\min \sum_{i:(i,j)\in\mathcal{A}_s} (D_{ij}^s(x_{ij}^s) + p_{ij}^s x_{ij}^s) \tag{5.3}$$

subject to

$$\textstyle\sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^s \geq R_s, \tag{5.4}$$

$$x_{ij}^s \geq 0, \qquad \forall\ i : (i,j) \in \mathcal{A}_s. \tag{5.5}$$

The bidding strategy of player j^s consists of two main components: bandwidth requests and price adjustments.

Bandwidth requests

If the bid prices p_{ij}^s's are given, the requested bandwidths at player j^s towards each of its upstream peers in overlay s, *i.e.*, x_{ij}^s, $\forall\ i : (i,j) \in \mathcal{A}_s$, can be optimally decided by solving the problem **Bidding $\mathbf{j^s}$**. This can be done efficiently with a water-filling approach, in which player j^s acquires the required streaming rate R_s by requesting from upstream peers that incur minimum marginal costs:

Let $f_j^s(x)$ denote the overall cost at player j^s, i.e., $f_j^s(x) = \sum_{i:(i,j)\in\mathcal{A}_s}(D_{ij}^s(x_{ij}^s)+p_{ij}^s x_{ij}^s)$. *The marginal cost with respect to x_{ij}^s is $\frac{df_j^s(x)}{dx_{ij}^s} = D_{ij}'^s(x_{ij}^s) + p_{ij}^s$. Beginning with $x_{ij}^s = 0$*

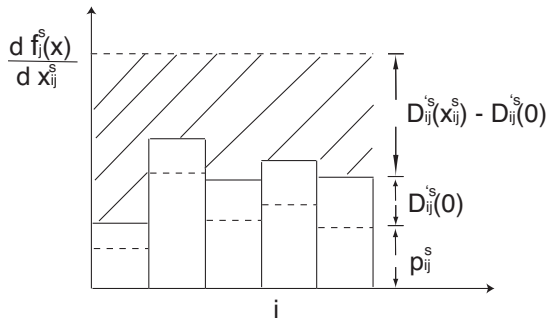


Figure 5.3: Bandwidth requesting strategy at player j^s: an illustration of the water-filling approach.

($\forall\ i : (i,j) \in \mathcal{A}_s$), the player identifies one x_{ij}^s that achieves the smallest marginal cost and increases the value of this x_{ij}^s. As $D_{ij}^s(x_{ij}^s)$ is strictly convex, $D_{ij}^{'s}(x_{ij}^s)$ increases with the increase of x_{ij}^s. The player increases this x_{ij}^s until its marginal cost is no longer the smallest. Then it finds a new x_{ij}^s with the current smallest marginal cost and increases its value. This process repeats until the sum of all x_{ij}^s's ($\forall\ i : (i,j) \in \mathcal{A}_s$) reaches R_s. □

The water-filling approach can be illustrated in Fig. 5.3, in which the height of each bin represents the marginal cost for player j^s to stream from each upstream peer i. To fill water at a total quantity of R_s into these bins, the bins with the lowest heights are flooded first, until all bins reach the same water level. Then the same water level keeps increasing until all the water has been filled in.

Theorem 1. *Given bid prices $p_{ij}^s,\ \forall\ i : (i,j) \in \mathcal{A}_s$, the water-filling approach obtains a unique optimal requested bandwidth assignment at player j^s,* i.e., *($x_{ij}^{s*},\ \forall\ i : (i,j) \in \mathcal{A}_s$), which is the unique optimal solution to the problem* ***Bidding*** $\mathbf{j^s}$.

Proof: Let $x_{ij}^{s*},\ \forall\ i : (i,j) \in \mathcal{A}_s$ be an optimal solution to the problem **Bidding $\mathbf{j^s}$** in (5.3). Introducing Lagrange multiplier λ for the constraint in (5.4) and $\nu = (\nu_i,\ \forall\ i : (i,j) \in \mathcal{A}_s)$ for the constraints in (5.5), we obtain the KKT conditions for the problem **Bidding $\mathbf{j^s}$**

as follows (pp. 244, [23]):

$$\sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^{s*} \geq R_s,$$
$$\lambda^* \geq 0,$$
$$x_{ij}^{s*} \geq 0, \nu_i^* \geq 0, \ \forall\ i:(i,j)\in\mathcal{A}_s,$$
$$\lambda^*(R_s - \sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^{s*}) = 0, \tag{5.6}$$
$$x_{ij}^{s*}\nu_i^* = 0, \ \forall\ i:(i,j)\in\mathcal{A}_s, \tag{5.7}$$
$$D_{ij}^{'s}(x_{ij}^{s*}) + p_{ij}^s - \lambda^* - \nu_i^* = 0,$$
$$\forall\ i:(i,j)\in\mathcal{A}_s. \tag{5.8}$$

For $x_{ij}^{s*} > 0$, we have $\nu_i^* = 0$ from (5.7). Then from (5.8), we derive the marginal cost with respect to x_{ij}^{s*}, $\frac{df_j^s(x^*)}{dx_{ij}^s} = D_{ij}^{'s}(x_{ij}^{s*}) + p_{ij}^s = \lambda^*$. Since D_{ij}^s is strictly convex and twice differentiable, the inverse function of $D_{ij}^{'s}$, i.e., $D_{ij}^{'s-1}$, exists and is continuous and one-to-one. Then we have $\forall\ i:(i,j)\in\mathcal{A}_s$)

$$x_{ij}^{s*} = \begin{cases} 0 & \text{if } \lambda^* < D_{ij}^{'s}(0) + p_{ij}^s, \\ D_{ij}^{'s-1}(\lambda^* - p_{ij}^s) & \text{if } \lambda^* \geq D_{ij}^{'s}(0) + p_{ij}^s. \end{cases} \tag{5.9}$$

In deriving the optimal solution which achieves a same marginal cost value λ^* for all the positive x_{ij}^{s*}'s, we always increase the smallest marginal cost $D_{ij}^{'s}(x_{ij}^s)+p_{ij}^s$ by increasing the corresponding x_{ij}^s. In this way, we are increasing the marginal costs towards the same value of λ^*. As $\lambda^* > 0$, we derive that x_{ij}^{s*}'s satisfy $R_s - \sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^{s*} = 0$ based on (5.6). Therefore, this water-filling process continues until R_s is used up, i.e., $\sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^{s*} = R_s$, by which time we obtain the unique optimal solution defined by (5.9). □

Price adjustments

We next address how each player is to determine the bid price to each of its desirable upstream peers. A price adjustment scheme is designed for this purpose, by which each player tactically adjusts its prices in participating auctions based on whether its bandwidth requirement is achieved in the previous bidding round.

When a player j^s first joins an overlay, it initiates bid prices p_{ij}^s's towards all desired upstream peers to 0. Then it calculates the current optimal requested bandwidth assignment with the *water-filling approach*, and sends its bids to the upstream peers. After upstream peer i allocates its upload capacity with the *allocation strategy*, it sends allocated bandwidth values to corresponding players. Upon receiving an allocated bandwidth, player j^s increases the corresponding bid price if its "demand" is higher than the "supply" from the upstream peer, and otherwise decreases the price. Meanwhile, it recomputes its requested bandwidth assignment for all its upstream peers with the water-filling approach. Such price adjustment is carried out in an iterative fashion, until the player's bandwidth requests may all be granted at respective upstream peers if it is to bid the new prices in the next round.

Using the water-filling approach as a building block, the price adjustment scheme is summarized in the *bidding strategy* to be carried out by player j^s in each round of its participating auctions, as presented in Table 5.1.

The intuition behind the bidding strategy is that, each player places different bid prices to different upstream peers, considering both the streaming cost and the overall demand at each upstream peer. If the streaming cost is low from an upstream peer, the player is willing to pay a higher price and strives to acquire more upload bandwidth from this peer. On the other hand, if the bandwidth competition at an upstream peer

Table 5.1: Bidding strategy at player j^s

Input

–(p_{ij}, x_{ij}): bids submitted in previous bidding round

–allocated bandwidth a_{ij}^s in previous bidding round from all upstream peers i, $\forall\ i : (i,j) \in \mathcal{A}_s$.

Adjust prices and bandwidth requests

Repeat

(a) For each upstream peer i

– If $x_{ij}^s > a_{ij}^s$, increase price p_{ij}^s by a small amount δ;

– If $x_{ij}^s \leq a_{ij}^s$ and $p_{ij}^s > 0$, decrease price p_{ij}^s by δ.

(b) Adjust requested bandwidth assignment (x_{ij}^s, $\forall\ i : (i,j) \in \mathcal{A}_s$) with the water-filling approach.

(c) For each upstream peer i

– Calculate new allocation a_{ij}^s that can be acquired from i if the current price p_{ij}^s is bid, based on Eqn. (5.2), with queried bids of some other players in the previous round of auction i.

Until: all requested bandwidths x_{ij}^s's, are to be achieved with current prices p_{ij}^s's, *i.e.*, $x_{ij}^s \leq a_{ij}^s$, $\forall\ i : (i,j) \in \mathcal{A}_s$, and prices p_{ij}^s's are the lowest possible to achieve it.

Submit new bids

Send new bids $b_{ij}^s = (p_{ij}^s, x_{ij}^s)$, $\forall\ i : (i,j) \in \mathcal{A}_s$, to respective upstream peers.

is intense such that the bidding cost becomes excessive, the player will forgo its price increases and request more bandwidths from other peers. At all times, the marginal cost of streaming from each upstream peer is kept the same, as achieved by the water-filling process.

We note that to calculate the new achievable allocation a_{ij}^s, player j^s needs to know bids placed by some of its opponents in the previous bidding round in auction i. Instead of asking upstream peer i to send all received bids, player j^s can query such information gradually only when necessary. If p_{ij}^s is to be increased, it asks for the bid of opponent $m^{s'}$

whose price $p_{im}^{s'}$ is immediately higher than p_{ij}^{s} in auction i. While p_{ij}^{s} is still below $p_{im}^{s'}$, player j^s's achievable bandwidth is unchanged; only when p_{ij}^{s} exceeds $p_{im}^{s'}$, its achievable bandwidth is increased by $a_{im}^{s'}$, and player j^s queries upstream peer i again for the bid containing the immediately higher price than the current value of p_{ij}^{s}. Similar bid inquiries can be implemented for the case that p_{ij}^{s} is to be reduced. In this way, the price adjustments can be achieved practically with little messaging overhead.

5.3 Game Theoretical Analysis

The distributed auction games in the coexisting streaming overlays are carried out in a repeated fashion, as these are *dynamic games*. They are correlated with each other as each player optimally places its bids in multiple auctions. A critical question is: *Does there exist a stable "operating point" of the decentralized games, that achieves efficient partition of network upload bandwidths?* We now seek to answer this question with game theoretical analysis.

We consider upload bandwidth competition in the entire network as one *extended* dynamic non-cooperative strategic game (referred to as G_{ext}), containing all the distributed correlated auctions. The set of players in the extended game can be represented as

$$\mathcal{I} = \{j^s,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S}\}. \tag{5.10}$$

The action profile taken by player j^s is a vector of bids, in which each component is the bid to place to one upstream peer. Formally, the set of action profiles for player j^s is defined as

$$\Gamma_j^s = \{B_j^s | B_j^s = (b_{ij}^s,\ \forall\ i : (i,j) \in \mathcal{A}_s),$$
$$b_{ij}^s = (p_{ij}^s, x_{ij}^s) \in [0, +\infty) \times [0, R_s], \sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^s \geq R_s\}. \tag{5.11}$$

Then, let B denote the bid profile in the entire network, *i.e.*, $B = (B_j^s,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S}) \in \times_{j,s}\Gamma_j^s$. The preference relation $\succsim_j^s$ for player j^s can be defined by the following overall cost function, which is the objective function in the problem **Bidding $\mathbf{j^s}$** in (5.3)

$$\text{Cost}_j^s(B) = \sum_{i:(i,j)\in\mathcal{A}_s} (D_{ij}^s(x_{ij}^s) + p_{ij}^s x_{ij}^s). \tag{5.12}$$

Therefore, we say two bid profiles $B \succsim_j^s B'$ if $\text{Cost}_j^s(B) \leq \text{Cost}_j^s(B')$.

Definition 1. *A bid profile B in the network, $B = (B_j^s,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S}) \in \times_{j,s}\Gamma_j^s$, is feasible if its bandwidth requests further satisfy upload capacity constraints at all the upstream peers, i.e., $\sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} x_{ij}^s \leq U_i,\ \forall\ i \in \mathcal{V} \cup \mathcal{N}$.*

When a bid profile is feasible, from the allocation strategy discussed in Sec. 5.2.1, we can see the upload bandwidth allocations will be equal to the requested bandwidths.

Using $\widetilde{B_j^s}$ to represent action profiles of all players other than player j^s in $\mathcal{I}$, *i.e.*, $\widetilde{B_j^s} = (B_m^k,\ \forall\ m^k \in \mathcal{I} \setminus \{j^s\})$, we have the following definition of Nash equilibrium.

Definition 2. *A feasible bid profile $B^* = (B_j^{s*},\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S})$ is a Nash equilibrium of the extended game $G_{ext}\langle\mathcal{I}, (\Gamma_j^s), (\succsim_j^s)\rangle$ if for every player $j^s \in \mathcal{I}$, we have $Cost_j^s(B_j^{s*}, \widetilde{B_j^{s*}}) \leq Cost_j^s(B_j'^s, \widetilde{B_j^{s*}})$ for any other feasible bid profile $B' = (B_j'^s, \widetilde{B_j^{s*}})$.*

We next show the existence of a Nash equilibrium for the extended game. We focus on feasible streaming scenarios as stated in the following assumption:

Assumption 1. *The total upload bandwidth in the P2P network is sufficient to support*

all the peers in all overlays to stream at required rates, i.e.*, there exists a feasible bid profile in the P2P network.*

Theorem 2. *In the extended game* $G_{ext}\langle \mathcal{I}, (\Gamma_j^s), (\succsim_j^s)\rangle$ *in which distributed auctions are dynamically carried out with the allocation strategy in (5.2) and the bidding strategy in Table 5.1, there exists a Nash equilibrium under Assumption 1.*

Proof: Define $\mathcal{B}$ to be the set of all possible bid profiles in the entire network, *i.e.*, $B \in \mathcal{B}$ and $\mathcal{B} \subset \times_{j,s}\Gamma_j^s$. From the definition of Γ_j^s in (5.11), we know $\mathcal{B}$ is convex and x_{ij}^s's are bounded. In addition, under Assumption 1, all peers in all overlays can obtain enough streaming bandwidths, and thus their bid prices to upstream peers will not be increased infinitely in the respective auctions. Therefore, all prices p_{ij}^s's in a possible bid profile are bounded, *i.e.*, $\exists\ \bar{p} > 0, p_{ij}^s \in [0, \bar{p}],\ \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$. Altogether, we derive that $\mathcal{B}$ is a convex compact set.

The action profile for each player j^s — $B_j^s = (b_{ij}^s,\ \forall\ i : (i,j) \in \mathcal{A}_s)$ — can also be represented as $B_j^s = (P_j^s, X_j^s)$, where $P_j^s = (p_{ij}^s,\ \forall\ i : (i,j) \in \mathcal{A}_s)$ is the vector of bid prices toward all upstream peers of player j^s, and $X_j^s = (x_{ij}^s,\ \forall\ i : (i,j) \in \mathcal{A}_s)$ is the vector of requested bandwidths towards all upstream peers at player j^s.

The price adjustment strategy described in Table 5.1 defines a mapping function θ_j^s, from bid profile B in the previous bidding round, to new prices to bid by player j^s, *i.e.*, $P_j^s = \theta_j^s(B)$.

Given price vector P_j^s, Theorem 1 gives that the water-filling approach uniquely decides the best requested bandwidth assignment X_j^s at player j^s. This mapping from price vector P_j^s to requested bandwidth vector X_j^s can be defined as function $X_j^s = \varphi_j^s(P_j^s)$.

Let $g_j^s(B) = (\theta_j^s(B), \varphi_j^s(\theta_j^s(B))$ be the mapping function from bid profile B in the

previous bidding round to a new action profile at player j^s. Let $g(B) = (g_j^s(B),\ \forall\ j \in \mathcal{N}_s, s \in \mathcal{S})$. Therefore, $g(B)$ is a point-to-point mapping from $\mathcal{B}$ to $\mathcal{B}$. The Nash equilibrium is a fixed-point of this mapping. We next show the existence of such a fixed-point.

We first show φ_j^s is a continuous mapping. Given P_j^s, X_j^s is the optimal solution of **Bidding j^s**. Therefore, φ_j^s is defined by (5.9) in the proof of theorem 1. Since D_{ij}^s is strict convex and twice differentiable, we know $D_{ij}^{'s}$ is continuous. Thus based on the water-filling process, we know the optimal marginal cost λ^* is continuous on P_j^s. Furthermore, as $D_{ij}^{'s-1}$ is continuous too, from (5.9), we derive X_j^s is continuous on P_j^s, *i.e.*, φ_j^s is continuous.

We next show θ_j^s is a continuous mapping from $\mathcal{B}$ to vector space of bid prices P_j^s at player j^s. Let p_{ij}^s be the bid price player j^s places to upstream peer i in the previous bidding round, and q_{ij}^s be the new bid price after the price adjustment defined by θ_j^s. Without loss of generality, we simplify our proof by showing that a small disturbance of the previous price p_{ij}^s to $p_{ij}^{'s} = p_{ij}^s + \epsilon, \epsilon > 0, \epsilon \to 0$ results in little disturbance at new price q_{ij}^s, *i.e.*, letting $q_{ij}^{'s}$ denote the new price corresponding to $p_{ij}^{'s}$, we have $q_{ij}^{'s} \to q_{ij}^s$. We divide our discussions into 2 cases, and first give a result to be used in the discussions: If p_{ij}^s is increased to $p_{ij}^{'s}$ and all other bid prices at player j^s remain unchanged, the corresponding requested bandwidth to upstream peer i is decreased, *i.e.*, $x_{ij}^{'s} \leq x_{ij}^s$. This can be directly obtained from the water-filling process used to solve **Bidding j^s**.

We now investigate the two cases:

A) At upstream peer i, there is no bid price from other players right between p_{ij}^s and $p_{ij}^{'s}$, *i.e.*, there does not exist p_{im}^k, such that $p_{ij}^s \leq p_{im}^k \leq p_{ij}^{'s}$.

Starting the price adjustment described in Table 5.1 from p_{ij}^s and $p_{ij}^{'s}$ respectively, we consider two sub cases: (i) If p_{ij}^s is to be reduced as $x_{ij}^s \leq a_{ij}^s$, we know $p_{ij}^{'s}$ is to be

reduced too, since $x_{ij}^{'s} \leq x_{ij}^{s}$ but $a_{ij}^{'s} \geq a_{ij}^{s}$ (due to $p_{ij}^{'s} > p_{ij}^{s}$). When $p_{ij}^{'s}$ is reduced, it will soon reach p_{ij}^{s}, and its following adjustments will be the same as those for p_{ij}^{s}. (ii) Similarly, if p_{ij}^{s} is to be increased, it will soon reach $p_{ij}^{'s}$ and their following adjustments will be the same. In both cases, we have for the new prices $q_{ij}^{'s} \rightarrow q_{ij}^{s}$.

B) At upstream peer i, there is a bid price from another player which lies right between p_{ij}^{s} and $p_{ij}^{'s}$, *i.e.*, $\exists\ p_{im}^{k}$, such that $p_{ij}^{s} \leq p_{im}^{k} \leq p_{ij}^{'s}$.

We again discuss three sub cases: (i) If p_{ij}^{s} is to be reduced due to $x_{ij}^{s} \leq a_{ij}^{s}$, $p_{ij}^{'s}$ is to be reduced too, since $x_{ij}^{'s} \leq x_{ij}^{s}$ but $a_{ij}^{'s} \geq a_{ij}^{s}$ (due to $p_{ij}^{'s} \geq p_{im}^{k} \geq p_{ij}^{s}$). During $p_{ij}^{'s}$'s adjustments, its value is continuously decreased, passing p_{im}^{k} and reaching p_{ij}^{s}. Then its following adjustments will be the same as those for p_{ij}^{s}. (ii) If p_{ij}^{s} is to be increased due to $x_{ij}^{s} > a_{ij}^{s}$ and $p_{ij}^{'s}$ is to be reduced as $x_{ij}^{'s} \leq a_{ij}^{'s}$, they will both stop at a same value near p_{im}^{k}. (iii) If both p_{ij}^{s} and $p_{ij}^{'s}$ are to be increased, p_{ij}^{s}'s value will be continuously increased to pass p_{im}^{k} and reach p_{ij}^{s}. Then their following adjustments will be the same. In all cases, the new prices $q_{ij}^{'s} \rightarrow q_{ij}^{s}$.

Therefore, based on the continuity of θ_{j}^{s} and φ_{j}^{s}, we derive that the mapping $g_{j}^{s}(B) = (\theta_{j}^{s}(B), \varphi_{j}^{s}(\theta_{j}^{s}(B))$ is continuous. Thus, $g(B) = (g_{j}^{s}(B),\ \forall\ j \in \mathcal{N}_{s}, s \in \mathcal{S})$ is a continuous mapping from $\mathcal{B}$ to itself. Based on Brouwer Fixed Point Theorem, any continuous mapping of a convex compact set into itself has at least one fixed point, *i.e.*, $\exists\ B^{*} = g(B^{*}) \in \mathcal{B}$. In addition, the fixed point B^{*} must be a feasible profile, as otherwise the adjustments of prices and requested bandwidths do not converge. Therefore, the fixed point B^{*} is a Nash equilibrium of the extended game. □

The next theorem shows that at equilibrium, the upload bandwidth allocation in the network achieves the minimization of the global streaming cost.

Theorem 3. *At Nash equilibrium of the extended game* $G_{ext}\langle \mathcal{I}, (\Gamma_j^s), (\succsim_j^s)\rangle$*, upload bandwidth allocation in the network achieves streaming cost minimization, as achieved by the following global streaming cost minimization problem:*

$$\min \sum_{s\in\mathcal{S}} \sum_{j\in\mathcal{N}_s} \sum_{i:(i,j)\in\mathcal{A}_s} D_{ij}^s(y_{ij}^s) \tag{5.13}$$

subject to

$$\textstyle\sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} y_{ij}^s \le U_i, \quad \forall\ i \in \mathcal{V}\cup\mathcal{N}, \tag{5.14}$$

$$\textstyle\sum_{i:(i,j)\in\mathcal{A}_s} y_{ij}^s \ge R_s, \qquad \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S}, \tag{5.15}$$

$$y_{ij}^s \ge 0, \qquad \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}. \tag{5.16}$$

Proof: We prove by showing that at equilibrium, the KKT conditions satisfied by the equilibrium bid profile $B^* = ((p_{ij}^{s*}, x_{ij}^{s*}),\ \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S})$ are the same as KKT conditions for the global streaming cost minimization problem in (5.13).

At equilibrium, given p_{ij}^{s*}'s, the requested bandwidths at each player j^s, x_{ij}^{s*}, $\forall\ i : (i,j) \in \mathcal{A}_s$, are the optimal solution to the problem **Bidding j^s**, and are also the same as allocated bandwidths from respective upstream peers. Therefore, altogether, we know the bandwidth allocations in the entire network, x_{ij}^{s*}, $\forall\ (i,j) \in \mathcal{A}_s$, $\forall\ s \in \mathcal{S}$, solve the following optimization problem:

$$\min \sum_{s\in\mathcal{S}} \sum_{j\in\mathcal{N}_s} \sum_{i:(i,j)\in\mathcal{A}_s} (D_{ij}^s(x_{ij}^s) + p_{ij}^{s*} x_{ij}^s) \tag{5.17}$$

subject to

$$\sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^s \geq R_s, \quad \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S}, \tag{5.18}$$

$$x_{ij}^s \geq 0, \qquad \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}, \tag{5.19}$$

and also satisfy upload capacity constraints at all the upstream peers:

$$\sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} x_{ij}^{s*} \leq U_i, \quad \forall\ i \in \mathcal{V} \cup \mathcal{N}. \tag{5.20}$$

Introducing Lagrange multiplier $\lambda = (\lambda_j^s,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S})$ for the constraints in (5.18) and $\nu = (\nu_{ij}^s,\ \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S})$ for constraints in (5.19), we obtain the KKT conditions for the optimization problem in (5.17) as follows:

$$\begin{aligned}
\sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} x_{ij}^{s*} &\leq U_i,\ \forall\ i \in \mathcal{V} \cup \mathcal{N},\\
\sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^{s*} &\geq R_s,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S},\\
x^* \geq 0, \lambda^* \geq 0, \nu^* &\geq 0,\\
\lambda_j^{s*}(R_s - \sum_{i:(i,j)\in\mathcal{A}_s} x_{ij}^{s*}) &= 0,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S},\\
x_{ij}^{s*}\nu_{ij}^{s*} &= 0,\ \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}, \qquad (5.21)\\
D_{ij}^{'s}(x_{ij}^{s*}) + p_{ij}^{s*} - \lambda_j^{s*} - \nu_{ij}^{s*} &= 0,\\
\forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}. &\qquad (5.22)
\end{aligned}$$

For $x_{ij}^{s*} > 0$, we have $\nu_{ij}^{s*} = 0$ from (5.21), and $D_{ij}^{'s}(x_{ij}^{s*}) + p_{ij}^{s*} = \lambda_j^{s*}$ from (5.22). Since D_{ij}^s is strictly convex and twice differentiable, the inverse function of $D_{ij}^{'s}$, i.e., $D_{ij}^{'s-1}$,

exists. Then we have $\forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$

$$x_{ij}^{s*} = \begin{cases} 0 & \text{if } \lambda_j^{s*} < D_{ij}^{'s}(0) + p_{ij}^{s*}, \\ D_{ij}^{'s-1}(\lambda_j^{s*} - p_{ij}^{s*}) & \text{if } \lambda_j^{s*} \geq D_{ij}^{'s}(0) + p_{ij}^{s*}. \end{cases} \tag{5.23}$$

Similarly, for the global streaming cost minimization problem in (5.13), introducing Lagrange multiplier $q = (q_i,\ \forall\ i \in \mathcal{V} \cup \mathcal{N})$ for the constraints in (5.14), $\lambda = (\lambda_j^s,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S})$ for the constraint in (5.15) and $\nu = (\nu_{ij}^s,\ \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S})$ for constraints in (5.16), we obtain the following KKT conditions:

$$\begin{aligned} \sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} y_{ij}^{s*} \leq U_i,\ \forall\ i \in \mathcal{V} \cup \mathcal{N}, \\ \sum_{i:(i,j)\in\mathcal{A}_s} y_{ij}^{s*} \geq R_s,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S}, \\ y^* \geq 0, q \geq 0, \lambda^* \geq 0, \nu^* \geq 0, \\ q_i^*(\sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} y_{ij}^{s*} - U_i) = 0,\ \forall\ i \in \mathcal{V} \cup \mathcal{N}, \\ \lambda_j^{s*}(R_s - \sum_{i:(i,j)\in\mathcal{A}_s} y_{ij}^{s*}) = 0,\ \forall\ j \in \mathcal{N}_s,\ \forall\ s \in \mathcal{S}, \\ y_{ij}^{s*}\nu_{ij}^{s*} = 0,\ \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}, \\ D_{ij}^{'s}(y_{ij}^{s*}) + q_i^* - \lambda_j^{s*} - \nu_{ij}^{s*} = 0, \\ \forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}. \end{aligned} \tag{5.24, 5.25}$$

And similarly, we can obtain $\forall\ (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$

$$y_{ij}^{s*} = \begin{cases} 0 & \text{if } \lambda_j^{s*} < D_{ij}^{'s}(0) + q_i^*, \\ D_{ij}^{'s-1}(\lambda_j^{s*} - q_i^*) & \text{if } \lambda_j^{s*} \geq D_{ij}^{'s}(0) + q_i^*. \end{cases} \tag{5.26}$$

To show the two sets of KKT conditions are actually the same, we first demonstrate that at each upstream peer i, the equilibrium bid prices from all competing players that are allocated non-zero bandwidths are the same, *i.e.*, $\exists\ t_i^*, p_{ij}^{s*} = t_i^*$ if $x_{ij}^{s*} > 0$, $\forall\ j : (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$. This can be illustrated as follows: If a player in auction i is paying a price higher than some other player who is also allocated non-zero bandwidth, the former can always acquire more bandwidth from the later with a price lower than its current price. Thus at equilibrium, when no one can unilaterally alter its price, all players must be paying the same price.

In addition, we know from the price adjustment described in Table 5.1 that if upstream peer i's upload capacity is large enough to satisfy all bandwidth requests, the corresponding bid prices in auction i can all be lowered down to 0. Therefore, at equilibrium, if $U_i > \sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} x_{ij}^{s*}$, we have $p_{ij}^{s*} = t_i^* = 0,\ \forall\ j : (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$. Thus we know t_i^* satisfies

$$t_i^*(\sum_{s\in\mathcal{S}} \sum_{j:(i,j)\in\mathcal{A}_s} x_{ij}^{s*} - U_i) = 0,\ \forall\ i \in \mathcal{V} \cup \mathcal{N}. \tag{5.27}$$

From these results and comparing (5.27)(5.22) with (5.24)(5.25) respectively, we can derive $t_i^* = p_i^*,\ \forall\ i \in \mathcal{V} \cup \mathcal{N}$, and the two sets of KKT conditions are actually the same. Therefore, the equilibrium solution in (5.23) is the same as the optimal solution to the global streaming cost minimization problem in (5.26). Thus Theorem 3 is proven. □

From the proof of Theorem 3, we can derive the following corollary:

Corollary. *At Nash equilibrium, the bid prices to each upstream peer i from all competing players that are allocated non-zero bandwidths are the same,* i.e., $\exists\ t_i^*, p_{ij}^{s*} = t_i^*$ *if* $x_{ij}^{s*} > 0$, $\forall\ j : (i,j) \in \mathcal{A}_s,\ \forall\ s \in \mathcal{S}$.

This corollary can also be intuitively illustrated: If a player in auction i is paying a price higher than some other player who is also allocated non-zero bandwidth, the former can always acquire more bandwidth from the latter with a price lower than its current price. Thus at equilibrium, when no one can unilaterally alter its price, all players must be paying the same price.

5.4 Convergence in Practical Scenarios

In contrast to existing game theoretic approaches that require synchronous play, we next show that our auction strategies can be practically applied in realistic asynchronous and dynamic P2P streaming networks, and prove the actual convergence of the dynamic auctions to the desirable Nash equilibrium.

5.4.1 Asynchronous play

In a practical P2P network, peers are inherently asynchronous with different processing speeds. Besides, with various message passing latencies, bids and allocated bandwidth updates may arrive at each upstream or downstream peer at different times. All these make each auction completely asynchronous. A practical deployment of the game theoretical strategies should be able to practically handle such asynchronous game play.

In our design, bids and allocation updates are passed by messages sent over TCP, such that their arrival is guaranteed. The strategies are to be carried out practically in each auction in the following fashion:

Allocation. At each upstream peer, starting from the last time it sends out allocation updates, the upstream peer collects new bids until it has received them from all its

existing downstream peers in all the overlays it participates in, or a timeout value, T, has passed since the previous allocation, whichever is shorter. Then the upstream peer allocates its upload bandwidth again with the allocation strategy discussed in (5.2): for those downstream peers whose bids it has received, it uses the new bids; for those slow ones that it has not heard from in this round, it uses the most recent bids from them. Then the upstream peer sends allocation updates to all the downstream peers whose allocation has been changed, and starts a new round of execution.

Bidding. At each downstream peer in each streaming overlay, since its last bidding round, it waits for bandwidth allocation until all allocated bandwidth updates have arrived from all its requested upstream peers, or time T has passed since the last time it placed all the bids, whichever is shorter. Then it adjusts prices towards those upstream peers from which it has received allocation updates, retains its previous prices towards those it has not heard from in this round, and recalculates its new bids to all the upstream peers, using the bidding strategy in Table 5.1. It then submits new bids, that are different from the ones submitted previously, to the respective upstream peers.

While we have established the existence of Nash equilibrium of the distributed auctions in Theorem 2, we have not yet addressed another critical question: *Can the Nash equilibrium,* i.e.*, the stable operating point that achieves optimal bandwidth allocation among peers in different overlays, be actually reached with such dynamic asynchronous play of the auction games?* We now seek to justify such a convergence, based on the following assumptions:

Assumption 2. *a) Each downstream peer in each overlay will communicate its new bids to its upstream peers within finite time (until it has acquired the required streaming bandwidth for the overlay at the lowest possible prices); b) Each upstream peer will*

communicate latest allocation updates to its downstream peers within finite time.

Theorem 4. *Under Assumption 1 and 2, the asynchronous distributed auctions converge to the Nash equilibrium, whose existence is established in Theorem 2.*

Proof: We first note the following property of the extended auction game, as modeled in Sec. 5.3:

Claim 1. The total allocated upload bandwidth in the entire network, $U_{alloc} = \sum_{s\in\mathcal{S}} \sum_{(i,j)\in\mathcal{A}_s} a_{ij}^s$, grows monotonically during the asynchronous play of the extended auction game.

The truth of the above claim lies in the fact that once a unit of upload bandwidth is allocated during the auction, it remains allocated throughout the rest of the game; *i.e.*, a unit of allocated bandwidth may switch its ownership from a downstream peer i in overlay s_1 to another downstream peer j in overlay s_2, but will never become idling again.

We further know U_{alloc} is bounded above by the total upload bandwidth in the network, $U_{all} = \sum_{i\in\mathcal{V}\cup\mathcal{N}} U_i$. Since U_{alloc} is increasing and upper-bounded, it therefore converges. Let U_{alloc}^* be its convergence value.

We now prove the theorem by contradiction. We assume that the extended auction game does not converge and runs for an infinitely long time. By Assumption 2, we know that there must exist peers that do not obtain sufficient bandwidth and thus bid infinitely often with updated prices. In this case, U_{alloc}^* must be smaller than the aggregated bandwidth demand at all peers in all the overlays, as otherwise peers stop bidding and the auction terminates.

When the total allocated bandwidth U_{alloc}^* is not sufficient to satisfy the bandwidth requirement at all the peers, based on our price adjustment strategy, we know the bid

prices at all upstream peers will be growing unbounded. In this case, there must not exist an upstream peer that still has spare upload bandwidth, *i.e.*, all upload bandwidth in the network has been allocated. We thus derive $U^*_{alloc} = U_{all}$. Therefore, U_{all} is smaller than the total bandwidth demand at all the peers in all the overlays, which contradicts with assumption 1. Thus the extended auction game must converge. In addition, the extended auction game must converge to its Nash Equilibrium (since peers would otherwise continue bidding), which achieves streaming cost minimization based on Theorem 3. □

5.4.2 Peer dynamics

Peer asynchrony aside, the inherent dynamics of realistic P2P network further leads to dynamics of auction participants. The players in each auction may change dynamically, due to new peers joining the network, existing peers joining another overlay or switching upstream peers due to content availability, or peer failures and departures; a distributed auction may start or close due to the arrival or departure of an upstream peer. With slightly more extra effort, our asynchronous deployment can readily adapt to such dynamics.

At the downstream side, when a peer newly joins an auction at an upstream peer, *e.g.*, in the cases of arrival of a new downstream or upstream peer, it initializes its bid price to 0, computes requested bandwidth together with its prices to other upstream peers, and then forwards its bid to the upstream peer. In the case that one of its upstream peers fails or departs, the downstream peer can detect it based on the broken or closed connections. Then it may exclude the upstream peer from its bandwidth request calculation.

At the upstream side, when an upstream peer receives the bid from a new peer, it

immediately incorporates the downstream peer into its bandwidth allocation. When it detects the failure or departure of a downstream peer based on the broken or closed connections, the upstream peer allocates upload bandwidth to the remaining peers only in a new round, excluding the departed peer from the auction game.

When peer dynamics are present in the network, the dynamic auction game progresses and strives to pursuit the optimal bandwidth allocation in the latest overlay topology. When peers continue to join and leave the overlays, the optimal bandwidth allocation naturally becomes a moving target. When such dynamics stop and overlay topology stabilizes, if the overall bandwidth supply is sufficient in the current topology, by results in Theorem 2, 3 and 4, we know there exists a Nash Equilibrium which achieves global streaming cost minimization, and the auction game converges to such an optimal streaming topology.

5.5 Performance Evaluation

Using simulations under real-world asynchronous settings in practical scenarios, the focus of our performance evaluation is to show that, as an outcome of our proposed auction strategies, coexisting overlay topologies can fairly share network bandwidth, evolve under various network dynamics, and can be prioritized.

The general realistic settings for our forthcoming experiments are as follows: Each network includes two classes of peers, 30% Ethernet peers with 10 Mbps upload capacities and 70% ADSL/Cable modem peers with heterogeneous upload capacities in the range of $0.4 - 0.8$ Mbps. Streaming servers in a network are Ethernet peers as well. We use delay-bandwidth products to represent streaming costs (M/M/1 delays), with streaming cost functions in the form of $D^s_{ij} = x^s_{ij}/(C_{ij} - x^s_{ij})$. Here, C_{ij} is the available

overlay link bandwidth, chosen from the distribution of measured capacities between PlanetLab nodes [4]. In asynchronous play of the auction games, the timeout value for an upstream/downstream peer to start a new round of bandwidth allocation/requested bandwidth calculation, T, is set to 1 second.

5.5.1 Limited visibility of neighbor peers

In Assumption 1 of our game theoretical analysis in Sec. 5.3, we assume that upload capacities in the network are sufficient to support all the peers to stream at required rates. This is generally achievable when the neighbor list maintained at each peer contains a lot of other peers in each overlay it participates in. However, in practical scenarios, each peer only has knowledge of a limited number of other peers, much smaller than the size of the network. We first study the convergence and optimality of the proposed strategies in such practical cases with asynchronous play of the auctions. As known neighbors constitute possible upstream peers in the streaming, the neighbor list at a peer is henceforth referred to as the *upstream vicinity* of the peer.

In our experiments, peers in upstream vicinity of each peer are randomly selected by a bootstrapping server from the set of all peers in the same overlay. The actual upstream peers at which each peer bids for bandwidth are decided by their content availability during streaming. Specifically, we seek to answer the following questions: First, what is the appropriate size of the upstream vicinity, such that the auctions converge and the required streaming bandwidth can be achieved at all peers in an overlay? Second, if the upstream vicinity is smaller, do the peers need to bid longer before the auction games converge? Finally, how different is the resulting optimal topologies when auction strategies are used with upstream vicinities of various sizes, with respect to streaming

cost?

Evaluation. We investigate by experimenting in networks with 100 to 10,000 peers with various sizes of upstream vicinities. In each network, we now consider a single overlay, with one server serving a 1 Mbps media stream to all peers.

Fig. 5.4 illustrates the outcome of our distributed auction strategies, either when they converge, or when a maximum bidding time, 2 seconds, has been reached. In the latter case, we assume the games have failed to converge, as there exist peers that cannot achieve the required streaming rate with their current size of upstream vicinities. Decreasing the size of upstream vicinities from $n-1$ where n is the total number of peers in each network, we discover that with $20-30$ peers in the upstream vicinity, the games can still converge and the required streaming rate can still be achieved at all peers in most networks, as shown in Fig. 5.4(A) and (B). Fig. 5.4(B) further reveals that convergence is always achieved rapidly (in a few seconds) in all networks with different sizes of upstream vicinities, as long as these games converge at all with a particular upstream vicinity size. A careful observation exhibits that the auction games take slightly longer to converge in larger networks. For a fixed network size, with larger upstream vicinity, a peer may spend more time in receiving bandwidth allocation and computing bandwidth requests, but carry out fewer bidding rounds before it acquires the required streaming rate. Therefore, the total time to convergence remains similar with different upstream vicinity sizes when the auctions converge, and only distinguishes itself in larger networks when they fail to converge.

Fig. 5.4(C) compares the optimality of resulting topologies in terms of their global streaming costs computed with the allocated bandwidths. Although each resulting topology achieves streaming cost minimization with respect to its own input mesh topology,

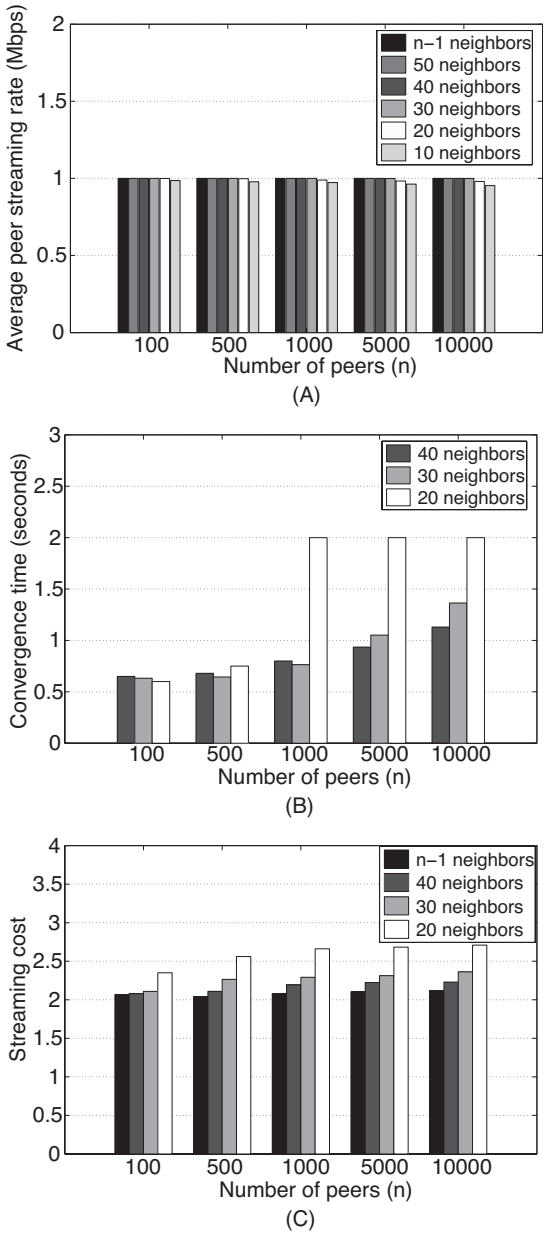


Figure 5.4: Outcomes of distributed auctions in networks of different sizes, and with various sizes of upstream vicinities.

the global streaming cost is less when the input topology is denser with larger upstream vicinities. However, compared to the ultimate minimum streaming cost achieved when upstream vicinities contain all other peers in the overlay, the cost experienced by using

upstream vicinities of a much smaller size (30) is only 10% higher.

Summary. From these observations, it appears that the appropriate size of upstream vicinities is relatively independent of network sizes, and the bandwidth allocation converges quickly in most cases. Both are good news when our auction strategies are to be applied in realistic large-scale networks. Based on results in this section, in our following experiments, the upstream vicinity size at each peer is set to 30.

5.5.2 The case of multiple coexisting overlays

We now proceed to study how our game strategies resolve the bandwidth competition among multiple coexisting streaming overlays. In particular, how does the topology of each overlay evolve, if coexisting overlays are started in the network? Do multiple coexisting overlays fairly share network bandwidth, and experience similar streaming costs?

Evaluation 1. We introduce more and more streaming overlays onto a 1000-peer network: At the beginning, all peers participate in one overlay and start to bid for their streaming bandwidths. Then every 10 seconds, the peers join one more new streaming overlay. To clearly show the effects of an increasing number of coexisting overlays on the allocated streaming bandwidth of each existing overlay, the required streaming rates for all overlays are set to the same 1 Mbps.

Fig. 5.5 illustrates the evolution of the average peer streaming rate in each overlay over time, when 5 overlays are sequentially formed in the network. We can see when there are up to 3 overlays coexisting in the network, the upload capacities in the network are sufficient for each overlay to achieve their required streaming rate. When there are 4 or 5 overlays, the capacities become insufficient to support all the overlays.

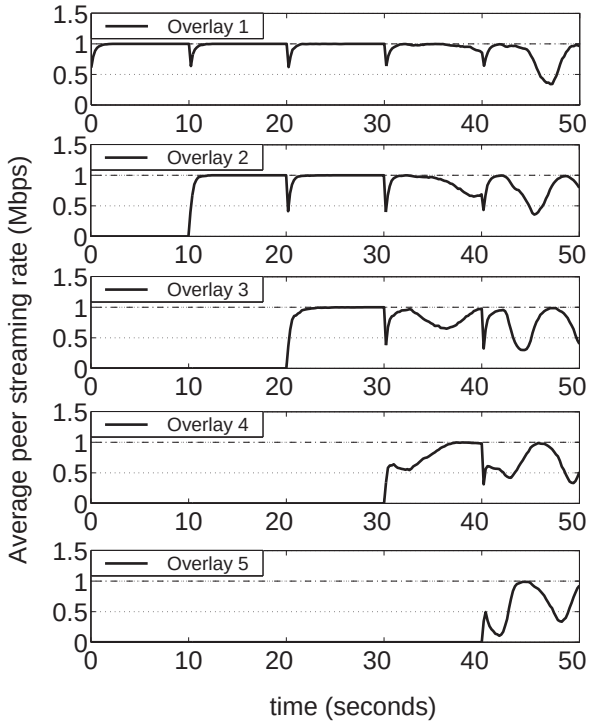


Figure 5.5: The evolution of peer streaming rate in multiple coexisting overlays with an increasing number of overlays over time.

In the former case with $1-3$ overlays, every time a new overlay is formed, the previous equilibrium bandwidth allocation across overlays is disturbed, and the games quickly converge to a new equilibrium, in which each overlay achieves the required streaming rate again. In addition, the costs experienced by coexisting overlays when their topologies stabilize are shown in Fig. 5.6. We observe both streaming and bidding costs are very similar across the multiple coexisting overlays.

In the latter case with $4-5$ overlays in the network, Fig. 5.5 shows that the games fail to converge, and the streaming bandwidth obtained by each overlay fluctuates over time. We observed during the experiment that peers in each overlay bid higher and higher prices at their upstream peers, but were nevertheless unable to acquire the required streaming rate. Similar bandwidth deficits can be observed in all coexisting overlays from Fig. 5.5.

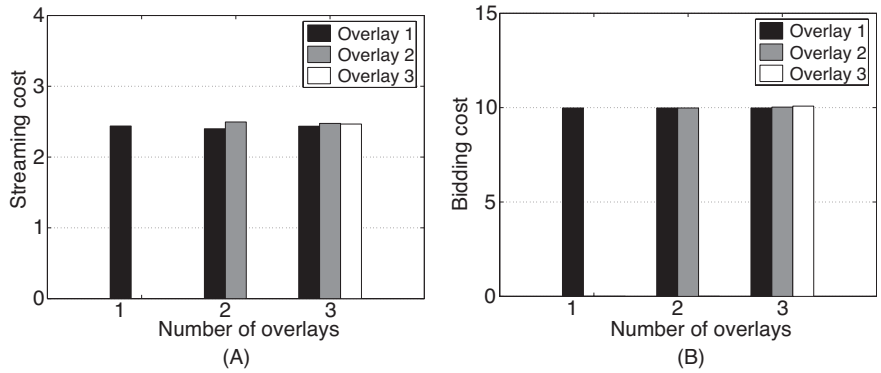


Figure 5.6: A comparison of costs among multiple coexisting overlays.

In practical P2P applications, some streaming overlays might expect to receive better service quality than others. For example, live streaming of premium television channels should enjoy a higher priority and better quality than regular ones. Since our game strategies can achieve fairness among various overlays (as observed from Fig. 5.5 and Fig. 5.6), we wonder if it is further possible to introduce a practical prioritization strategy in our games, such that differentiated service qualities can be provided to different overlays.

In our previous experiment, we have observed that overlays fairly share bandwidth for a simple reason: Peers in different overlays are not constrained by a bidding *budget*, and they can all raise bid prices at will to acquire more bandwidth from their desired upstream peers, which leads to relative fair bandwidth allocation at the upstream peers. Motivated by such insights, we introduce a budget-based strategy to achieve service differentiation, by offering higher budgets to peers in higher priority overlays. To introduce such budgets, we only need to make the following minor modification to the bidding strategy proposed in Sec. 5.2.2:

When a peer j joins a streaming overlay s, it obtains a bidding budget W_s from its bootstrapping server. Such a budget represents the "funds" peer j can use to acquire

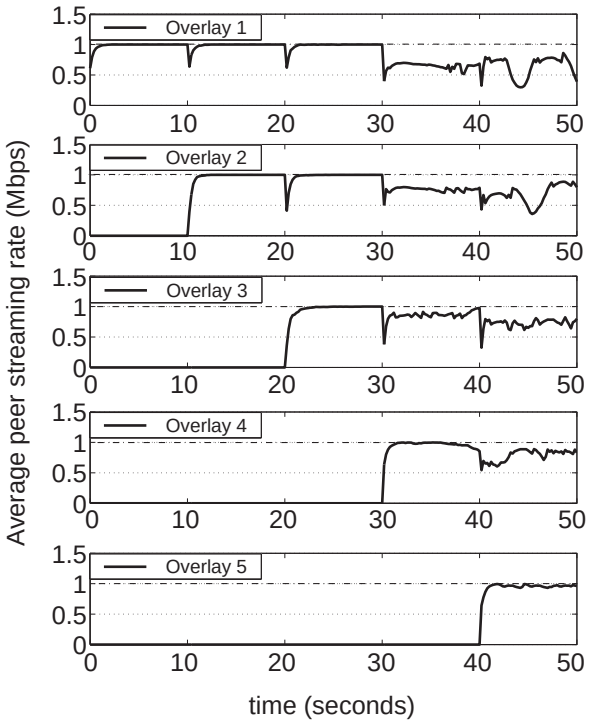


Figure 5.7: The evolution of peer streaming rate for multiple coexisting overlays with different budgets, and with an increasing number of overlays over time.

bandwidth in overlay s, and its total bidding costs to all upstream peers cannot exceed this budget, i.e., $\sum_{i:(i,j)\in\mathcal{A}_s} p_{ij}^s x_{ij}^s \leq W_s$. *All peers in the same overlay receive the same budget, and the bootstrapping server assigns different levels of budgets to different overlays based on their priorities. During its price adjustments in overlay s, peer j may only increase its bid prices if the incurred total bidding cost does not exceed W_s.*

Evaluation 2. Applying the budget-based bidding strategy, we perform the previous experiment again and show our new results in Fig. 5.7 and Fig. 5.8. The budgets assigned to peers in overlay 1 to 5 range from low to high.

Comparing Fig. 5.7 with Fig. 5.5 in the cases when 1 to 3 overlays coexist, we see that overlays can still achieve their required streaming rate within their budgets. However, when comparing Fig. 5.8 to Fig. 5.6(A), we observe that the streaming costs are

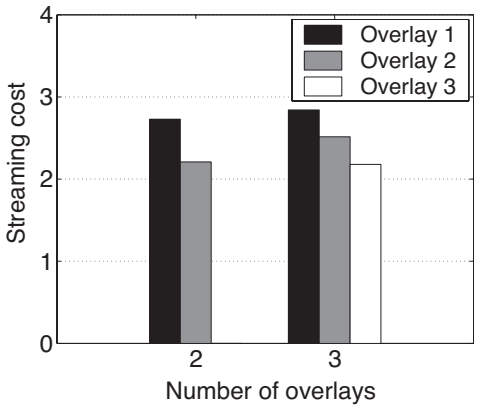


Figure 5.8: A comparison of streaming costs among multiple coexisting overlays with different budgets.

differentiated across overlays, *i.e.*, overlays with larger budgets are able to achieve lower streaming cost than those with smaller budgets. This is because the former can afford to pay higher prices and thus eclipse the latter in auctions at their commonly desired upstream peers.

A further comparison between Fig. 5.7 and Fig. 5.5 (when 4 or 5 overlays coexist) shows that, when upload capacities become insufficient, the overlay with the highest budget, overlay 4 or overlay 5 in respective phases, always achieves the highest and most stable streaming rates, while those for overlays with smaller budgets become less sufficient and less stable.

Summary. We have observed that, no matter if upload capacities are sufficient or not, our game strategies achieve fair bandwidth sharing among multiple coexisting overlays. When overlays are able to achieve their required streaming rates, they also experience similarly costs, which further reveal their fair share of lower latency paths. Further, we show that by introducing budgets to our bidding strategy, we are able to differentiate service qualities among coexisting overlays.

5.5.3 Overlay interaction under peer dynamics

In the following set of experiments, we study how coexisting streaming overlays evolve with peer arrivals and departures, with respect to how the achieved streaming rates in each overlay vary in such dynamics. We investigate both cases that the overlays have or do not have differentiated budgets.

Evaluation. We simulate a dynamic P2P streaming network, in which 2 servers concurrently broadcast 4 different 60-minute live streaming sessions, at the streaming rate of 300 Kbps, 500 Kbps, 800 Kbps and 1 Mbps, respectively. Starting from the beginning of the live broadcasts, 1000 peers join the network following a Poisson process. The inter-arrival times follow an exponential distribution with an expected length of *INTARRIV* seconds. Upon arrival, each peer randomly selects 2 broadcast sessions and joins the respective overlays; then the peer stays in the network for a certain period of time, following an exponential lifetime distribution with an expected length of *LIFETIME* seconds. In this way, we simulate 4 dynamically evolving streaming overlays with approximately the same number of participating peers at any time. Similar to the previous experiments, each peer maintains about 30 neighbors in each overlay it participates in, and bids at upstream peers that have available blocks to serve it. All other settings of the experiments are identical to those in previous experiments. We monitor the achieved streaming rates in each dynamic overlay during the 60-minute broadcasts.

Fig. 5.9 shows the results achieved when the budget-based strategy is not applied. Setting *INTARRIV* and *LIFETIME* to different values, we have repeated the experiment, and made the following observations: With expected inter-arrival time of 1 second, 1000 peers have all joined the network in the first 10 minutes; peer arrivals last for 45 minutes when *INTARRIV* is 3 seconds. With an expected lifetime of 10 minutes, most peers

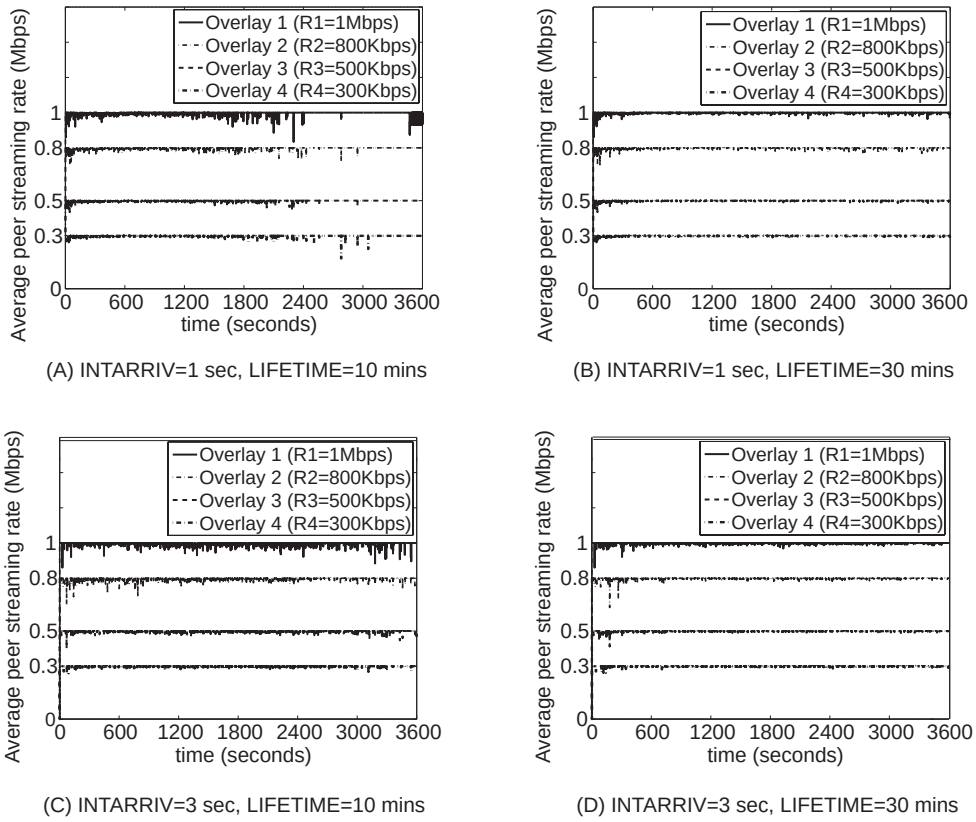


Figure 5.9: Achieved streaming rates for 4 coexisting overlays: under peer dynamics without budget

have left the network before the end of streaming; when *LIFETIME* is 30 minutes, approximately half of all the peers remain till the end.

Therefore, the most severe peer dynamics occurs when 1000 peers keep joining for 45 minutes, but have almost all left before 60 minutes, *i.e.*, the case shown in Fig. 5.9(C), which represents the highest level of fluctuations for the achieved streaming rates. A longer peer lifetime brings better overlay stability, which is illustrated by the smaller rate fluctuation in (B) and (D) of Fig. 5.9. A careful comparison of the fluctuation of the streaming rates across different overlays in Fig. 5.9 reveals slightly larger fluctuations for overlays with larger streaming rate requirements. This is because with our auction

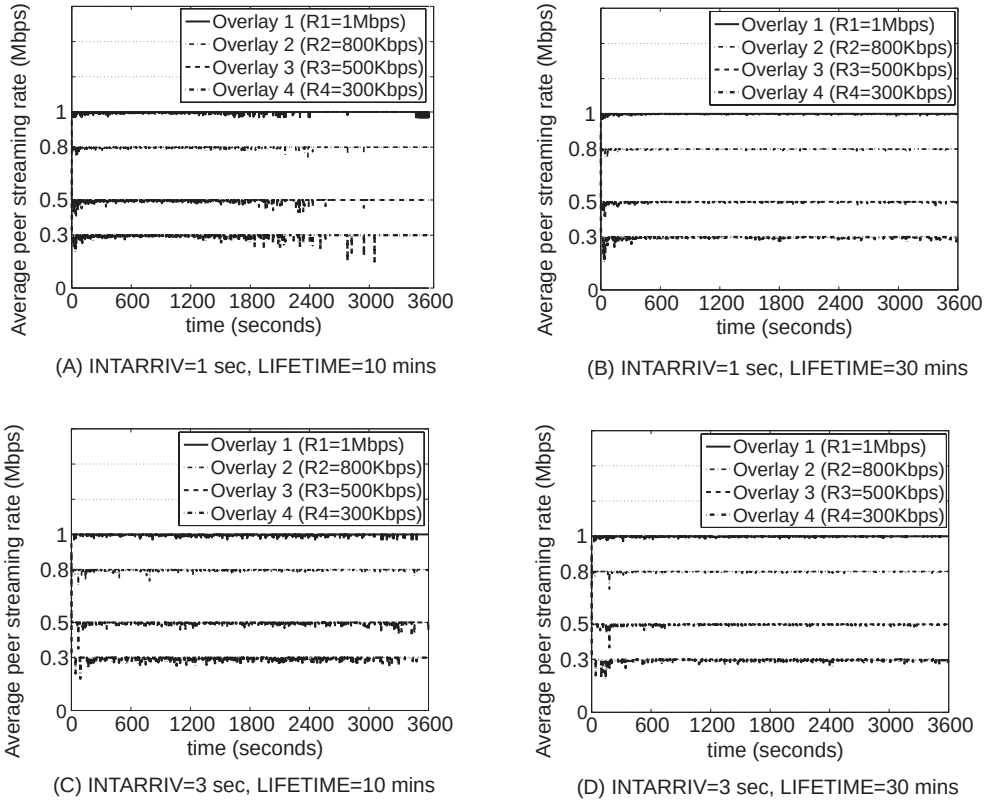


Figure 5.10: Achieved streaming rates for 4 coexisting overlays: under peer dynamics with different budgets.

strategies, different overlays fairly share upload capacities at common upstream peers, and the larger the required bandwidth is, the harder it is to achieve.

On the other hand, when overlays with higher rate requirement are prioritized with higher budgets, Fig. 5.10 shows a different outcome from that in Fig. 5.9. In Fig. 5.10, under all four interval settings, the prioritized high-rate overlays are always guaranteed more stable rates, while low-rate overlays experience more severe rate fluctuations.

Summary. We have clearly demonstrated the effectiveness of our auction strategies under high degrees of peer dynamics, which guarantee stable streaming bandwidth allocation for all overlays at all times during such dynamics. We have also shown that using

the budget-based bidding strategy, better streaming quality can be further provided for prioritized overlays.

5.5.4 Summary

In this chapter, we design conflict-resolving strategies for effective bandwidth sharing among multiple coexisting P2P streaming overlays. Our objective is to devise practical and completely decentralized strategies to allocate peer upload capacities, such that (1) the streaming rate requirement can be satisfied in each overlay, (2) streaming costs can be globally minimized, and (3) overlays fairly share available upload bandwidths in the network. Most importantly, we wish to achieve global optimality using localized algorithms. We use dynamic auction games to facilitate our design, and use game theory in our analysis to characterize the conflict among coexisting overlays. Different from previous work, our focus in applying game theory is not on reasoning about the rationality and selfishness of peers, nor on incentive engineering to encourage contribution, but to facilitate the design of the distributed strategies to achieve global properties. We finally show that our proposed algorithm adapts well to peer dynamics, and can be augmented to provision service differentiation.

Chapter 6

Measurement of a Large-Scale P2P Streaming Application

Commercial P2P live streaming applications have been successfully deployed in the Internet, with millions of users at any given time [88, 5, 10, 9, 6, 8, 2, 11]. The successful commercial deployment has made it possible to stream volumes of legal content to the end users, with hundreds of live media channels. Most of the recent P2P streaming applications adopt the common mesh-based streaming design, in which participating peers exchange available blocks of each media channel among each other.

Given the commercial success of mesh-based P2P streaming, it is an intriguing research challenge to explore and understand how their relatively simple designs actually behave in practice and dynamically evolve over a long period of time, in order to discover any design inefficiencies and possible ways for improvement. Towards these objectives, we have conducted an extensive measurement study of a large-scale P2P streaming application, in collaboration with UUSee Inc. [10], one of the leading P2P live streaming solution providers based in Beijing, China. In our study, we instrument the UUSee P2P

streaming application to collect large volumes of traces, which amount to almost a terabyte over a span of one year, involving millions of users, with a snapshot of the entire system every five minutes. As compared to all the existing P2P streaming measurement work discussed in Sec. 2.4, it is fair to claim that the scale of this work is unprecedented in P2P streaming research. Given this large volume of traces, we are able to thoroughly study many important facets of this practical large-scale P2P streaming system, which is not possible with the limited traces collected using crawling or passive sniffing, and have discovered many intriguing insights, which have never been revealed by previous measurement studies.

In this chapter, we overview the UUSee P2P streaming solution, discuss our detailed measurement methodology, and then present the basic global characteristics derived from the traces. In the following three chapters, we will discuss our in-depth study of the large-scale P2P streaming system in three different aspects based on the measurements.

6.1 UUSee P2P Streaming Solution

UUSee is the commercial application developed by UUSee Inc. [10], one of the leading P2P streaming solution providers based in Beijing, China. UUSee features exclusive contractual rights to most of the channels of CCTV, the official Chinese television network. When compared to PPLive (which is better known in the research community due to existing measurement studies), it features a rich collection of *legal* content, encoded and broadcast live over the Internet. The UUSee P2P streaming framework consists of a media encoding server, which encodes the media channels to high quality constant-rate streams around 400 Kbps using its own proprietary codec, and a large collection of dedicated streaming servers (approximately 150), which receive encoded media channels

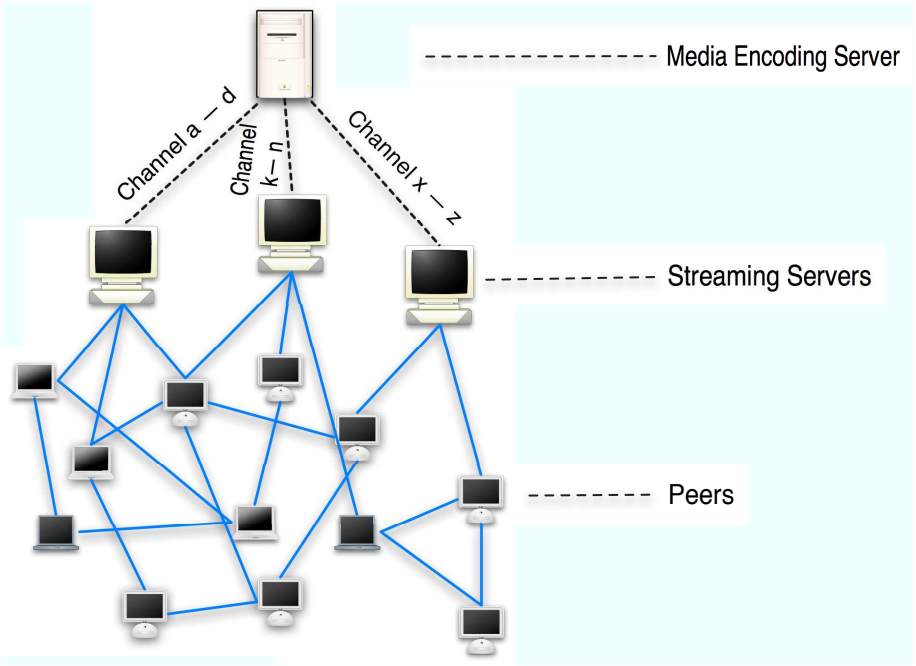


Figure 6.1: An illustration of UUSee P2P streaming network.

from the media encoding server and serve the P2P network composed of all the users of UUSee. An illustration of the UUSee streaming network is shown in Fig. 6.1. The streaming servers in UUSee are distributed in different regions in China and overseas (*e.g.*, Japan, U.S.), based on a rough estimation of the number of users in different areas. With its large collection of streaming servers around the world, UUSee simultaneously sustains over 800 media channels. With a growing user base, it routinely serves millions of users in any given day. Its Windows-based P2P streaming client represents one of the most popular downloads in this category.

Similar to most state-of-the-art mesh-based P2P streaming protocols, UUSee is designed with the principle of allowing peers to serve each other by exchanging blocks of data, that are received and cached in their local playback buffers. It employs a random peer selection strategy to assign initial *partners* to each peer, using central tracking

servers. The tracking servers maintain a list of existing peers in each media streaming channel during streaming. After a new peer joins a channel in UUSee, one of the tracking servers bootstraps it with an initial set of a small number of partners (up to 50), which are randomly selected from the list. The peer establishes TCP connections with these partners, and *buffer availability bitmaps* (*i.e.*, buffer maps) are exchanged periodically. During this process, it measures the TCP throughput of the connection, and also executes an estimation algorithm based on such measurements to predict the partner's availability to serve itself. It then ranks all known partners, and selects the best 30 peers from which it actually requests media blocks. A synchronous playback mechanism is employed in UUSee, by which each newly joined peer always starts to buffer the media blocks that are to be played 20 seconds later than the current playback time of the media channel at the media encoding server, and as thus, all peers in the channel are following a similar playback progress.

The buffer size at each peer in UUSee is 500 media blocks, and each block represents 1/3 second of media playback. The new peer starts the media playback from the first buffered block after 20 second, if a satisfactory buffering level has been reached during this time period. Otherwise, it will restart its initial buffering process for another 20 seconds, and then start the playback from the first block that has been buffered during this new 20 second period. Therefore, the initial start-up delay at the peers in UUSee is usually 20 seconds, and may be up to 40 seconds or 1 minute depending on the availability of media blocks in the network. During the playback at each peer, blocks to be played in the immediate future are continuously requested and cached in the playback buffer, and those that are not retrieved in time for playback are skipped. There is further a policy employed in the buffering process, that a peer will stop filling up its buffer when

the buffer has reached around 75% of its total size.

Beyond the random peer selection protocol, UUSee also incorporates a number of unique algorithms in its peer selection strategy, in order to maximally utilize peer upload bandwidth and to guarantee smooth media playback at the peers. During the initial start-up phase, each peer in UUSee employs an algorithm to estimate its maximum download and upload capacities. During the streaming process, each peer continuously estimates its aggregate instantaneous receiving and sending throughput from and to all its partners. If its estimated sending throughput is lower than its upload capacity for 30 seconds, it will inform one of the tracking servers that it is able to receive new connections from other peers. The tracking servers keep a list of peers that are able to accept new connections, and bootstrap new peers with existing peers that are randomly selected from this set.

The number of available blocks in the current playback buffer is used in UUSee to represent the current streaming quality of the peer, which is referred to as the *buffer count*. During the streaming process, in addition to exchanging new media blocks with each other, neighboring peers also recommend known peers to each other. The buffer count is used as an important criterion for such recommendations. When a peer i finds that another peer j has a low buffer count, *i.e.,* an insufficient number of blocks in its buffer, peer i will recommend its known peers with larger buffer counts. As a last resort, when a peer has a low buffer count for a sustained period of time, it will contact the tracking server again to obtain additional peers with better streaming qualities.

6.2 Trace Collection Methodology

To acquire a thorough understanding of this practical P2P streaming system, we collaborate with the UUSee development team to implement detailed measurement and reporting capabilities within its P2P client application. The client software on each peer in UUSee measures a wide range of performance metrics and protocol parameters during streaming, including the channel it is watching, its buffer count and advertised buffer map, its aggregate sending and receiving throughput, its total download and upload bandwidth capacities. In addition, for each active partner with which it has a live TCP connection, each peer also actively measures the number of sent or received blocks, as well as the maximum sending or receiving throughput of the TCP connection periodically.

6.2.1 Measurement method

While the collection of most performance metrics is straightforward, in what follows, we explain our measurement methods with respect to the download and upload capacities of each peer, and the maximum sending or receiving throughput along each TCP connection.

The download capacity of each peer is measured at the initial buffering stage of the peer, upon its first joining a streaming channel in the UUSee network. During this stage, the peer has no available blocks in its playback buffer, and can concurrently download from many supplying peers. In this case, its download bandwidth is largely saturated. Therefore, the download capacity of the peer is estimated as its maximum aggregate download rate at this initial buffering stage.

The upload capacity at each peer is measured upon its joining before the actual streaming starts, by setting up a temporary upload TCP connection with one of the nearest streaming servers. As we know, the upload bandwidth at each streaming server

is mostly saturated due to its main upload functions, while the download bandwidth is largely idle. Therefore, we utilize the spare download capacity of the streaming servers, and have each peer send a randomly generated probing flow to a streaming server that is nearest to itself. The duration of the flow should be long enough for its TCP throughput to become stable, usually in seconds. The streaming server measures the stabilized TCP throughput on this connection, which is then estimated as the upload capacity of the respective peer.

The maximum sending or receiving throughput along a live TCP connection is measured periodically in the following fashion: The measurement interval is further divided into 30-second sub intervals. In each sub interval, the time that is actually used to transmit media blocks is summarized, excluding the idle TCP periods. An average throughput is calculated with the number of bytes sent in the block transmission time divided by the length of this duration. The maximum throughput is then derived as the maximum of all such average throughputs within the measurement interval. Taking the average transmission throughput within 30 seconds, we smooth out the periods of very bursty TCP throughput; deriving the maximum of all such 30-second measurements, we aim to obtain the maximally achievable TCP throughput on the link between two peers.

6.2.2 Reporting mechanism

All the vital measurements are collected at each peer periodically and reported to a standalone trace server using UDP datagrams. Each report includes basic information such as the peer's IP address, the channel it is watching, its buffer count and advertised buffer map, its aggregate sending and receiving throughput, its total download and upload bandwidth capacities, as well as a list of all its partners, with their corresponding IP

addresses, TCP/UDP ports, number of blocks sent to or received from each partner, and current maximum sending/receiving throughput on each connection. The total size of each report is $3-4$ Kbytes.

In our trace collection, a new peer sends its initial report 20 minutes after it joins UUSee network, and sends subsequent reports periodically with an interval of 10 minutes in the first two months of our trace collection, which is expedited to 5 minutes later on. Considering the small size of each report, such reporting only introduces a very small amount of extra traffic during the streaming process, *i.e.*, no more than 100 bps per peer. The 20-minute initial report delay is enforced to ensure that the reports are sent by relatively long-lived peers in the channels. However, since each peer reports a large number of its active partners (up to hundreds), there is a high probability that transient peers may appear in the partner lists of at least one reporting peer as well.

We have commenced collecting these measurements to UUSee's trace server starting September 2006, by upgrading all existing UUSee clients to the new release that produces periodic reports. To visualize our traces, in Fig. 6.2, we illustrate an example topology snapshot constructed from the reports of three representative peers, taken at 10:08:45 a.m., September 5, 2006. Over a one-year span, we have collected up to 1 terabyte of traces on the trace server, constituting continuous-time snapshots of P2P streaming system throughout this period.

We note that the salient advantages of our trace collection lie not only in the unprecedentedly large volume of traces, but also at the completeness of the snapshots it captures. As compared to the snapshots collected using a crawling methodology, we are confident that our reports at each measurement time include more complete information from essentially all the peers in the system, and the constituted snapshots represent less

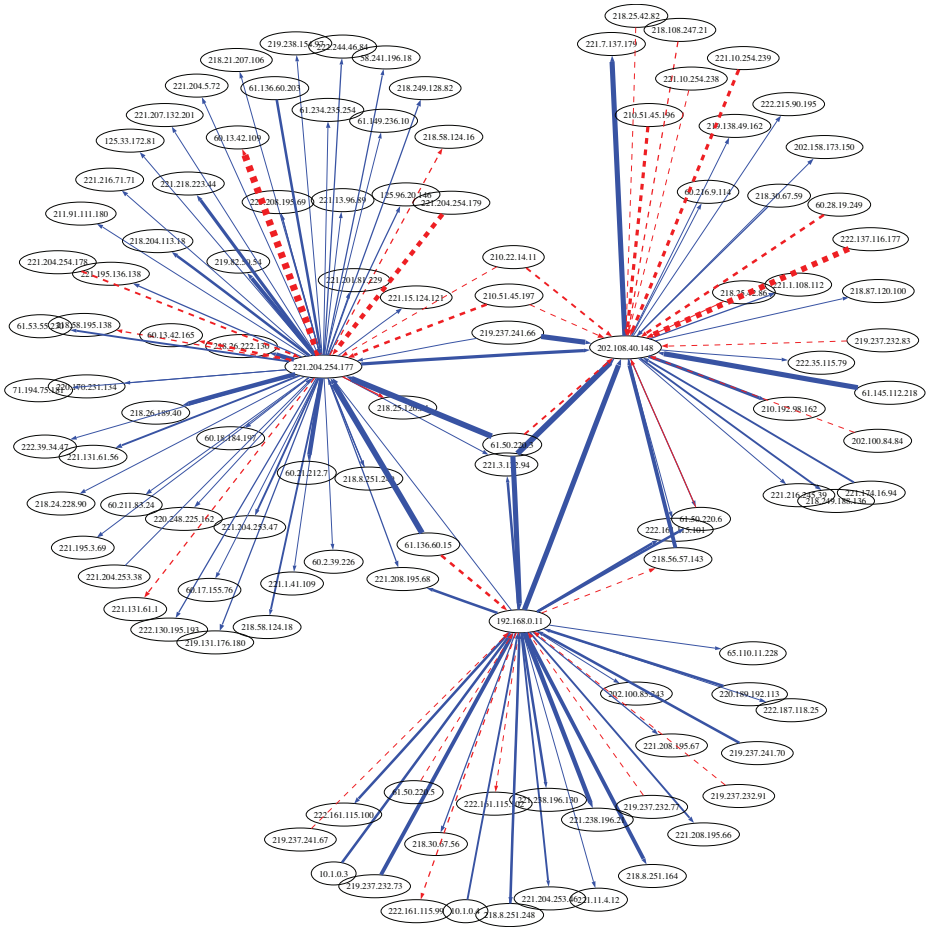


Figure 6.2: A snapshot involving three reporting peers and their active partners, visualized from traces collected at 10:08:45 a.m., September 5, 2006. While widths of both types of lines represent bandwidth, the dashed links have 10 times higher bandwidth per unit width than the solid ones.

distortion over the time domain as well.

6.3 Global Characterization

We now illustrate the basic global characteristics of the UUSee P2P streaming application, with respect to the scale of the streaming network, the distribution of peers, and the experienced streaming quality in the streaming channels. Due to the large volume of

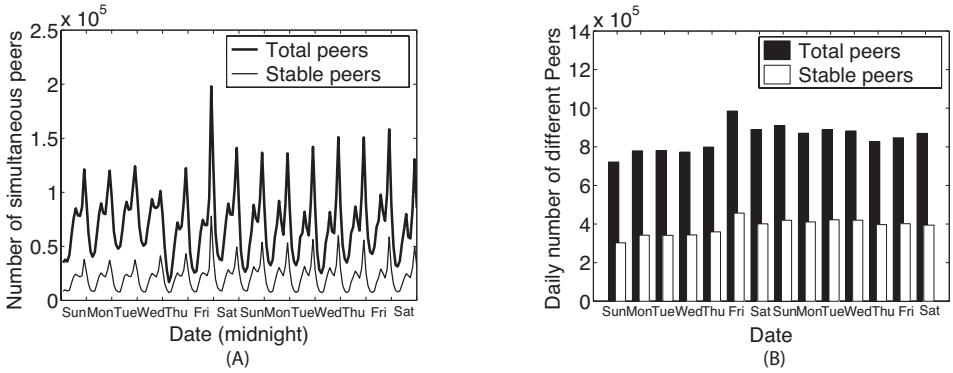


Figure 6.3: Daily peer number statistics from Sunday, October 1st, 2006 to Saturday, October 14th, 2006.

the traces, we will only present results obtained over representative weeks in our figures.

6.3.1 Overall number of simultaneous peers and flows

First of all, we are interested to investigate: (1) How many concurrent users are usually online in the UUSee streaming overlay? (2) What percentage do the stable peers (whose reports are received by the trace server) take in the peer population, as compared to transient peers (whose reports are not received)? To answer these questions, we summarize the IP addresses from which reports were received and recorded in the traces, and all the IP addresses that appeared in the traces, including IP addresses of peers that have reported and peers in their partner lists. The peer number statistics summarized from the traces collected from 12:00 a.m. October 1st, 2006 (GMT+8) to 11:50 p.m. October 14th, 2006 (GMT+8) are shown in Fig. 6.3(A).

The statistics indicate that there are around $100,000$ concurrent peers at any time in the UUSee streaming overlay. Comparing the number of stable peers to the total number of all peers, we discover that the former is asymptotically 1/3 of the later. For the daily evolution of peer number, there is a peak around 9 p.m., and a second peak around 1

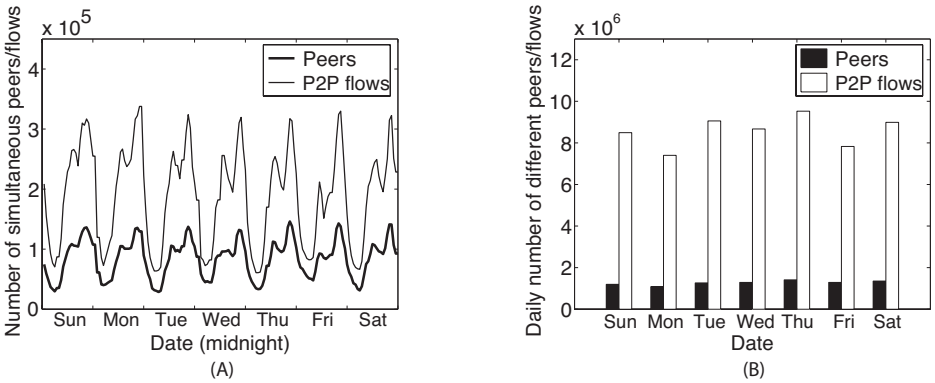


Figure 6.4: Daily peer/P2P flow number statistics from Sunday, December 17th, 2006 to Saturday, December 23, 2006.

p.m., which identify the same daily pattern as given in the study of PPLive [42]. Different from [42], we observe only a slight peer number increase over the weekend, considering the weekly variance trend.

To obtain a better idea of the scale of daily users of UUSee, we further summarize the number of distinct IP addresses that appeared in the traces on a daily basis in Fig. 6.3(B). The statistics show that UUSee serves up to 1 million different users each day.

All the above observations are further validated by the peer number statistics during a later time period, 12:00 a.m. December 17th, 2006 (GMT+8) to 11:50 p.m. December 23, 2006 (GMT+8), as shown in Fig. 6.4. In addition, Fig. 6.4 also summarizes the number of concurrent P2P flows among peers at each time, and the daily number of different P2P flows: There are on average $250,000$ active P2P flows at any time in the UUSee streaming network, and up to 10 million different flows on a daily basis.

Besides the regular daily peer/flow numbers, our trace study has revealed a few flash crowd scenarios during the trace period as well. For example, Fig. 6.3 shows a flash crowd scenario around 9 p.m. on October 6, 2006, which was caused by the broadcast of a celebration TV show for the mid-autumn festival in China. Another flash crowd

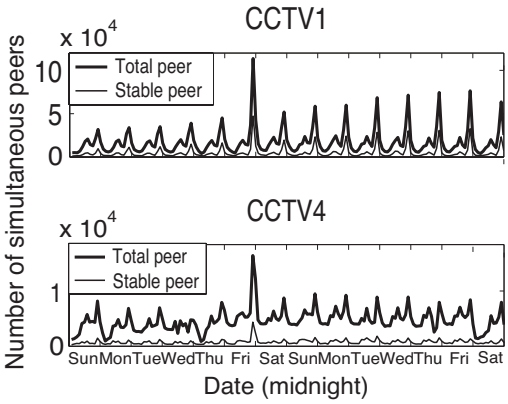


Figure 6.5: Daily peer number statistics in two representative channels: Sunday, October 1st, 2006 to Saturday, October 14th, 2006.

scenario was observed around 23 p.m. on February 17, 2007, due to the broadcast of the Chinese New Year celebration show, with 871,000 peers online in the UUSee streaming network and 2,271,000 streaming flows among the peers.

6.3.2 Number of simultaneous peers in two representative channels

As mentioned earlier, UUSee provides over 800 channels to the users. To investigate whether the observations we have made earlier regarding the global peer number evolution also apply to individual channels, we select two representative channels broadcast by UUSee, CCTV1 and CCTV4 (both from the official Chinese television network), where CCTV1 is among the most popular channels sustained in UUSee and CCTV4 has less popularity. Their peer number statistics are shown in Fig. 6.5. The different scales of the concurrent peer numbers clearly demonstrate the popularity difference of the two channels. Nevertheless, the evolutionary pattern of peer number in both channels is very similar to that in the global topology. In addition, both curves reflect a flash crowd

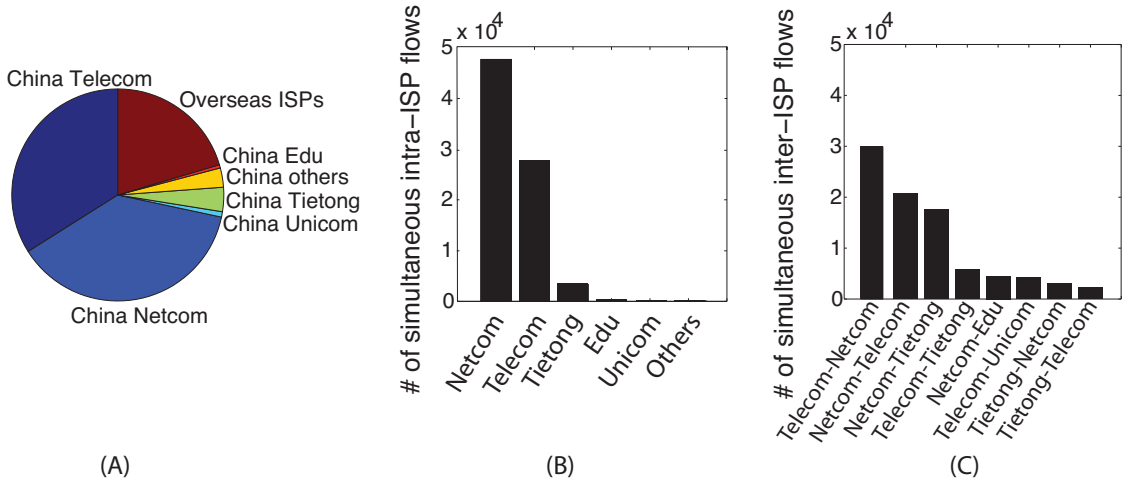


Figure 6.6: Peer/P2P flow number statistics for different ISPs.

scenario on October 6, 2006, as both channels broadcast the mid-autumn celebration TV show. The CCTV1 curve further exhibits a more distinctive daily peak on evenings, as is the prime time for China news broadcasting in the channel.

6.3.3 Number of simultaneous peers and flows in different ISPs

In our measurement study, we have also emphasized on the mapping of the abstract streaming topology to the real world scenario, with respect to the ISP and geographic area each peer is located at. For this purpose, we have obtained a mapping database from UUSee Inc. that translates ranges of IP addresses to their ISPs and geographic areas. For each IP address in China, the database provides the China ISP it belongs to and the city/province the user is located at; for each IP address out of China, it provides a general ISP code indicating foreign ISPs, and coarse geographic information of the continent the address lies in.

Using this mapping database, we have determined the ISP membership of simultaneous peers at any time. We have discovered that the ISP distribution of peers does not

vary significantly over time. Therefore, we only depict the averaged shares of peers in major ISPs over the trace period in Fig. 6.6(A).

It exhibits that the two largest nationwide ISPs in China, Netcom and Telecom, own the largest user shares in the UUSee P2P network. While the majority of UUSee users are in China, peers from overseas also take a significant 20%, and their percentage shows a rising trend as we observed in our investigation.

In addition, Fig. 6.6(B) summarizes the average number of concurrent P2P flows inside each major China ISP, and Fig. 6.6(C) illustrates the number of inter-ISP flows for ISP pairs that have more than 1000 concurrent flows in between. Again, the numbers of flows inside China Netcom and Telecom, and those for flows to and from these two ISPs dominate their respective categories.

6.3.4 Number of simultaneous peers and flows in different areas

Using the mapping database, we also determine the geographic distribution of simultaneous peers at any time. The distributions of IP addresses and P2P flows with respect to geographic regions are depicted in Fig. 6.7. With the majority of peers and links in China, those from North American, Europe and other Asian countries also have an adequate share. Another interesting discovery is that the evolutionary pattern of the number of concurrent North American users is similar to that of users in China. We identify the reason to be that the majority of UUSee users are watching CCTV channels, and their popular programs are broadcast live according to the same Beijing time (GMT+8).

.

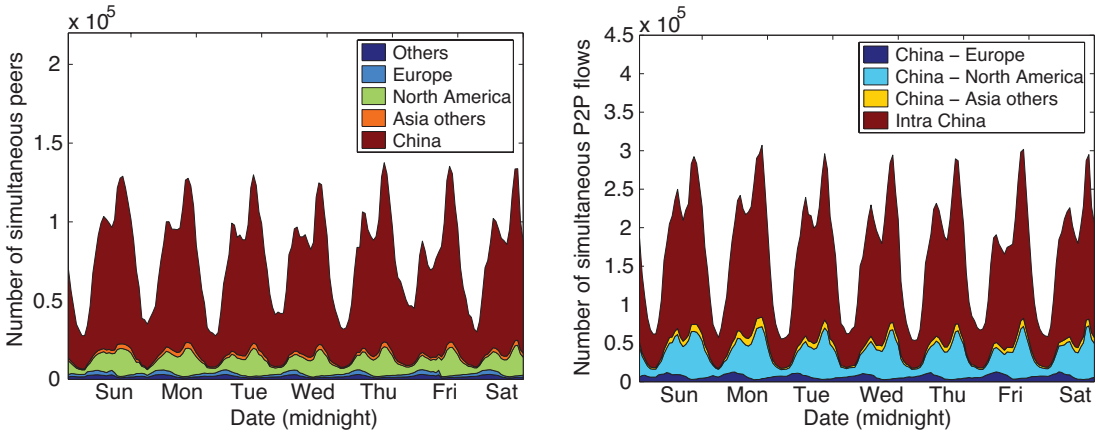


Figure 6.7: Peer/P2P flow number statistics for different geographic regions: Sunday, December 17th, 2006 to Saturday, December 23, 2006.

6.3.5 Different peer types

We next categorize peers into two classes based on their download capacities in the traces, and the fact that the download bandwidth of the fastest ADSL connection in China is at most 3 Mbps: (1) Ethernet peers, for those with download capacities higher than 384 KBps; (2) ADSL/cable modem peers, for the remainder. Fig. 6.8(A) exhibits the domination of ADSL/cable modem peer population. We infer the reason is that users tend to watch the broadcasts at home, and most of the residential high speed services are based on ADSL or cable modems. Nevertheless, Fig. 6.8(B) shows comparable shares of P2P flows in each category, which demonstrates the contribution of the limited number of Ethernet peers in uploading to many other peers.

6.3.6 Streaming quality

To explore the streaming quality of UUSee, we make use of two measurements collected at each peer in different channels — the number of available blocks in the current playback buffer (buffer count) and the aggregate instantaneous receiving throughput. With the

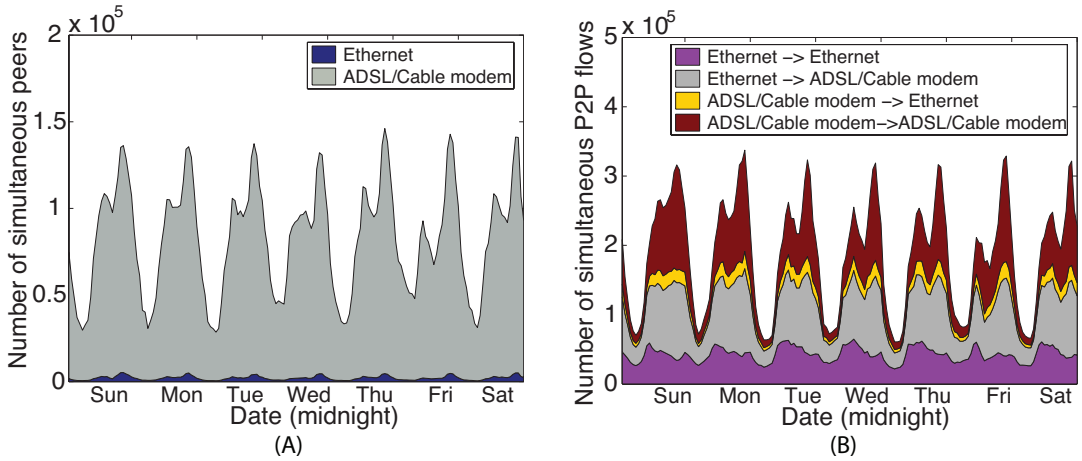


Figure 6.8: Peer/P2P flow number statistics in two peer type categories: Sunday, December 17th, 2006 to Saturday, December 23, 2006.

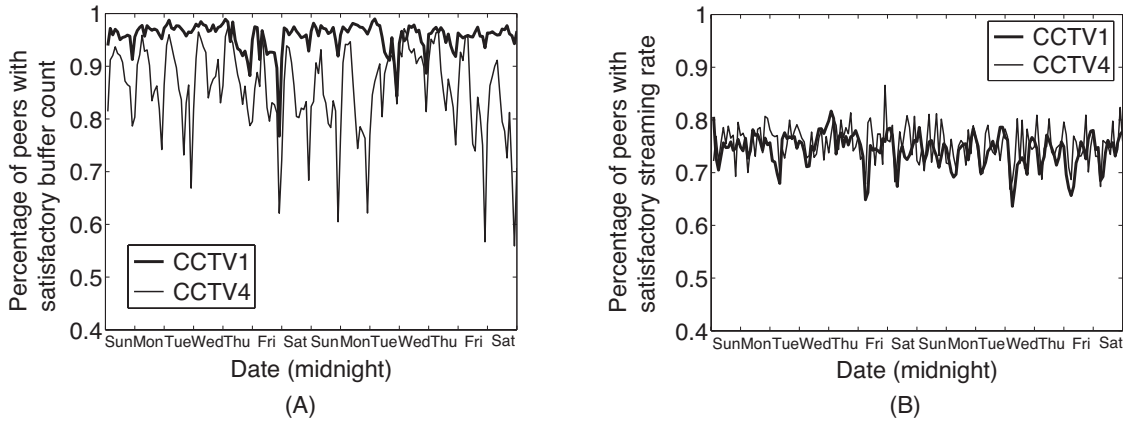


Figure 6.9: Percentage of peers with satisfactory streaming quality: Sunday, October 1st, 2006 to Saturday, October 14, 2006.

example of CCTV1 and CCTV4, Fig. 6.9(A) depicts the percentage of peers in both channels whose buffer counts are no lower than 75% of the total buffer size, and Fig. 6.9(B) shows the percentage of peers in both channels whose receiving throughput is higher than 90% of the channel streaming rate.

With respect to the buffer count metric, we observe that the buffering level for the popular channel CCTV1 is very encouraging, as for most of the time, around 95% of its

participating peers have a satisfactory playback buffer count. Considering the UUSee buffering policy discussed in Sec. 6.1 that a peer will stop buffering when its buffer has been roughly 75% full, this shows most of the peers in CCTV1 are having a "full" buffer for continuous playback. For the less popular channel CCTV4, the peer buffering level represents much larger fluctuations, and around 80% of its peers achieve satisfactory buffering for most of the times. Such a difference between popular and less popular channels exhibits that peers watching a popular channel can usually retrieve media blocks more easily as there are more supplying peers in the channel. A closer look at the plots further reveals that the buffering level is generally lower at the peak hours of a day, especially for the less popular channel.

In terms of the aggregate receiving throughput metric, we can see that around 3/4 of all viewers in both channels achieve satisfactory streaming rates, and no evident daily pattern is discovered for the evolution of the percentages. Comparing the two quality metrics, peer buffer count and aggregate receiving throughput, we would suggest that the buffer count metric better represents the actual streaming quality experienced at the peers, as it captures the block receiving quality in a recent period of time. The aggregate receiving throughput is measured instantaneously at each peer, and thus may be less accurate. Nevertheless, results from both metrics are quite positive, exhibiting that the UUSee peer selection protocol scales relatively well to a large number of peers.

6.4 Summary

In this chapter, we discuss our methodology of collecting a large volume of traces from a commercial P2P live streaming application. We also present our global characterization of the P2P application, with respect to the number of concurrent peers and P2P flows,

the distribution of peers/flows in different categories, and the streaming quality in the application. We have made a number of interesting observations: (1) There exist diurnal patterns in the evolution of the number of peers and the number of P2P flows in the system; (2) the majority of the system consist of ADSL/cable modem peers, but the limited number of Ethernet peers contribute significantly by uploading to many other peers; (3) more than 75% of all the peers in the system can achieve a satisfactory streaming quality, with a higher percentage for a more popular channel.

Chapter 7

Charting Large-Scale P2P Streaming Topologies

The practical success of current-generation P2P live streaming applications has validated important advantages of the mesh-based design over tree-based solutions: Mesh topologies achieve better resilience to peer dynamics, better scalability in flash crowd scenarios, more efficient use of bandwidth, as well as simplicity with respect to topology maintenance.

As mesh-based streaming topologies play an important role towards the recent commercial success of P2P live streaming, it would be an intriguing research topic to investigate and understand how the constructed mesh topologies actually *behave* in practice, dynamically *evolve* over time, and *react* to extreme scenarios such as huge flash crowds.

Unfortunately, although Internet topologies have been characterized extensively at the IP layer, there has been little literature on the discovery and analysis of P2P topologies in live P2P applications. As a well-known work on P2P topology characterization, Stutzbach *et al.* [79] explored topological properties of the file-sharing Gnutella network. However,

short-lived queries in Gnutella are fundamentally different from long-lived streaming sessions based on block exchanges among peers, which may well lead to different topological properties. In addition, like most of the existing measurement studies, their measurements rely on the methodology of crawling the Gnutella network, by which the speed of crawling plays a significant role in deciding the accuracy of the constructed topologies. The recent work of Vu *et al.* [83] has investigated two topological properties of PPLive, peer outdegree and clustering coefficient, but their work is based on limited information from a few hundreds of peers collected using crawling.

In this chapter, we seek to extensively explore and chart large-scale P2P live streaming meshes, and to gain in-depth insights and a complete understanding of their topological characteristics, using the instantaneous streaming topologies of the entire UUSee network presented by our traces. In particular, our study is primarily based on over 120 GB of traces that we have collected over a two-month period, from September to October, 2006, and our discoveries have been further validated using a more recent set of traces in 2007.

With emphasis on their evolutionary nature over a long period of time, we have utilized and extended classical graph measurement metrics — such as the degree, clustering coefficient, reciprocity and likelihood — to investigate various aspects of the streaming topologies at different times of the day, in different days in a week, and in flash crowd scenarios. To the best of our knowledge, this work represents the first complete and in-depth measurement study to characterize large-scale P2P live streaming topologies. Nevertheless, we also seek to compare our new insights with discovered topological properties from traditional file sharing P2P applications, to identify properties that are unique to mesh-based P2P streaming, and make comparisons with related results discussed in other P2P streaming measurement studies as well.

While we have analyzed the entire two-month trace period, in this chapter, we choose to show results from two representative weeks, from 12:00 a.m. October 1st, 2006 (GMT+8) to 11:50 p.m. October 14th, 2006 (GMT+8). We have observed that the selected periods include all typical scenarios that we wish to present, including the flash crowd on the evening of October 6, 2006. To validate the general applicability of our topological property discoveries, we also present representative topological characteristics from a more recent trace period, from 12:00 a.m. February 13th, 2007 (GMT+8) to 11:50 p.m. February 19th, 2007 (GMT+8), which includes another flash crowd scenario on February 17, 2007. Note that in all figures that present the temporal evolution of a metric, a small arrow is drawn to indicate the occurrence time of a flash crowd.

7.1 Degree Distribution

We first characterize the degree distributions at peers in the UUsee network. In our traces, each stable peer reports the IP addresses in its partner list, and also the number of blocks sent (received) to (from) each of the partners. With this information, we are able to categorize partners of each peer into three classes: (1) active supplying partner, from which the number of received blocks is larger than a certain threshold (*e.g.*, 10 blocks); (2) active receiving partner, to which the number of sent blocks is larger than the threshold; (3) non-active partner, in all the other cases.

With reports from stable peers in the traces, we investigate their degree distributions with respect to the number of active supplying partners (indegree), the number of active receiving partners (outdegree), and the total number of partners in the partner list including both active and non-active partners. Note that in a mesh network, it is common for a partner to be both a supplying partner and a receiving partner of a peer at the same

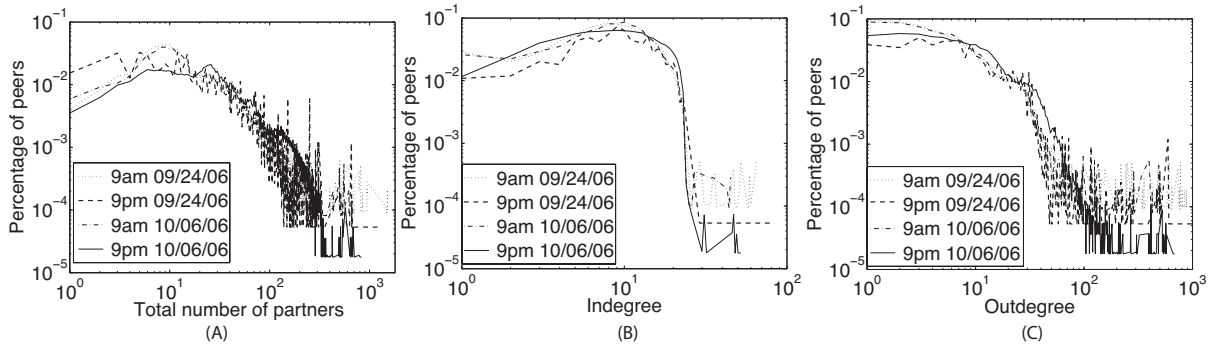


Figure 7.1: Degree distributions of stable peers in the global topology.

time. In this case, it is counted into both peer active indegree and active outdegree.

7.1.1 Degree distribution in the global topology

Most existing research on P2P topologies reported power-law degree distributions. In their studies on modern Gnutella topologies, Stutzbach *et al.* [79] pointed out that its degree distribution does not follow a power-law or two-segment power-law distribution, but has a spike around 30, as the Gnutella client software tries to maintain 30 neighbors for each peer. From Fig. 7.1(A), we observe that the distributions of total number of partners at the stable peers in the UUSee network do not follow power-law distributions either, with spikes whose corresponding degrees vary at different times. For the distributions observed in the morning, the spikes lie around a partner number of 10; for those observed in the daily peak hour, 9 p.m. at night, the spikes are located at larger values. During the flash crowd scenario around 9 p.m., October 6, 2006, the distribution peaks around 25. These reveal that at peak times, each peer is engaged with more partners. In addition, the results also exhibit that although each peer has an initial set of around 50 partners upon joining, the number of partners at different peers varies a lot during the streaming process, according to neighbor dynamics, the block availability at known

neighbors and the available upload capacity at each peer.

For the peer indegree distribution shown in Fig. 7.1(B), we observe spikes around 10, and the spike is at a slightly higher degree in the flash crowd scenario. For indegree distributions at all times, they drop abruptly when the indegree reaches about 23. According to the UUSee peer selection protocol, a peer only accepts new upload connections when it still has spare upload capacity, and thus the upload bandwidth on each upload link is guaranteed. Besides, during streaming, the aggregated download rate at each peer is limited by the streaming rate in UUSee. All these explain the observation that the number of active supplying peers at each peer, which guarantees satisfactory streaming rates, is relatively small in the UUSee overlay, as compared to other file sharing applications.

The peer outdegree distributions in Fig. 7.1(C) are closer to two-segment power-law distributions, with a joint point around degree 10. The curves for peak times exhibit a flatter first segment, which implies that peers have higher outdegrees when there are more requesting peers in the network.

7.1.2 Degree evolution in the global topology

We next show the evolution of average degrees of stable peers during the two-week period in Fig. 7.2(A). We observe that the averaged total number of partners peaks at the peak times, but the average peer indegree is consistently around 10. Given that UUSee does not explicitly impose such an upper bound for active incoming connections at each peer, we explain this phenomenon as follows. At peak times when a large number of peers coexist in the network, many peers will be able to offer help to others, and thus "volunteer" themselves at the tracking server, or get known by other peers when peers exchange their useful partner lists. This leads to the result that each peer knows a large number of

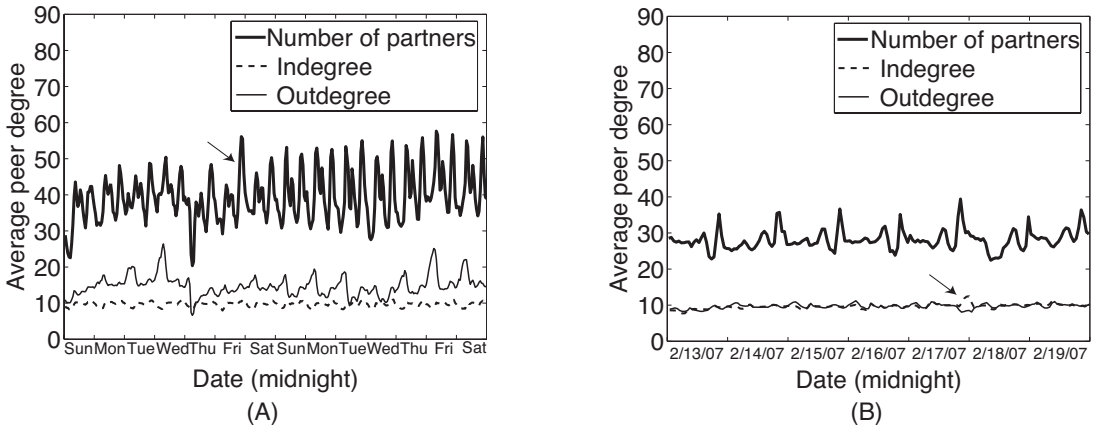


Figure 7.2: Evolution of average degrees for stable peers in the global topology. (A) From October 1, 2006 to October 14, 2006. (B) From February 13, 2007 to February 19, 2007.

other peers. Nevertheless, each peer does not actually need to stream from more peers to sustain a satisfactory streaming rate, as long as it streams from a few good ones.

To validate our discoveries with the more recent UUSee traces, we investigate the peer degree evolution during the week of February 13, 2007 to February 19, 2007, which also includes a flash crowd scenario due to the broadcast of Chinese New Year celebration show on the evening of February 17, 2007. From the results in Fig. 7.2(B), we can see the average number of partners and average outdegree per peer are smaller than those in Fig. 7.2(A), while the average indegree is at a similar level. The reduction of partner number may be attributed to the ISP upgrade of the access link bandwidth, which occurred in early 2007, so that peers can achieve satisfactory streaming rates without knowing many other peers. Nevertheless, the degree evolution patterns remain similar. In addition, we have also investigated the degree distributions at each specific time during the new trace period, which we have found represent similar shapes to those in Fig. 7.1, and are thus omitted for presentation.

In Vu *et al.*'s PPLive measurement study [83], they have derived an average node

outdegree within the range of 30 to 43. In their measurements, the outdegree at each peer includes all the partners that may retrieve from the peer, not necessarily only the ones that are actively streaming from it at each specified time, as how our outdegree is measured. Therefore, a fairer comparison would be to compare their results with our total number of partners, which are at a similar magnitude for both P2P streaming applications.

7.1.3 Intra-ISP degree evolution

To better understand the connectivity among peers in the same ISP and across different ISPs, we further investigate the active indegrees and outdegrees at each peer that are from and to peers in the same ISP. At each stable peer, we calculate the proportion of indegrees from partners in the same ISP to the total indegree of the peer, and the proportion of outdegrees toward partners in the same ISP to its total outdegree, respectively.

Fig. 7.3 plots the evolution of the intra-ISP degree percentage, averaged over all stable peers in the network at each time. From Fig. 7.3(A), we observe that the percentages for both indegrees and outdegrees are around 0.4. Considering that many ISPs coexist, this exhibits that the majority of active supplying/receiving partners of each peer are within the same ISP. Although UUSee does not take ISP membership into consideration when the tracking server assigns new partners to a peer and when neighboring peers exchange partners, this exhibits the "natural clustering" effects in the P2P streaming overlay over each ISP. The reason behind such clustering is that, as connections between peers in the same ISPs have generally higher throughput and smaller delay than those across ISPs, they are more inclined to be chosen as active connections.

In addition, Fig. 7.3(A) shows that the percentages for both indegree and outdegree

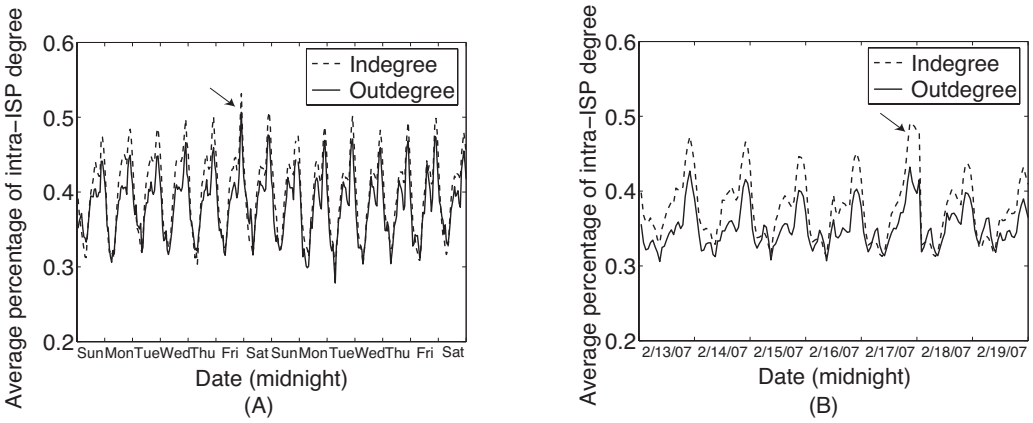


Figure 7.3: Evolution of average intra-ISP degrees for stable peers in the network. (A) From October 1, 2006 to October 14, 2006. (B) From February 13, 2007 to February 19, 2007.

peak at the daily peak hours and during the flash crowd scenario. This implies that each peer has more partner choices when the network is large, and it is always able to choose high throughput connections that are largely intra-ISP.

All the above observations are further validated by the investigation results using the more recent traces in February, 2007, as shown in Fig. 7.3(B). This exhibits the general applicability of our conclusions over a long period of time.

7.1.4 Intra-area degree evolution

Similarly, we further investigate the connectivity among peers in the same geographic location and across different areas. For IP addresses in China, they are in the same geographic area if they are located in the same province; for IP addresses out of China, they are grouped based on the continent they belong to. We compute the percentage of indegrees and outdegrees at each stable peer that are from and to peers in the same area at each time, and the evolution of averaged intra-area degree percentages (over all stable peers in the network at each time) is shown in Fig. 7.4.

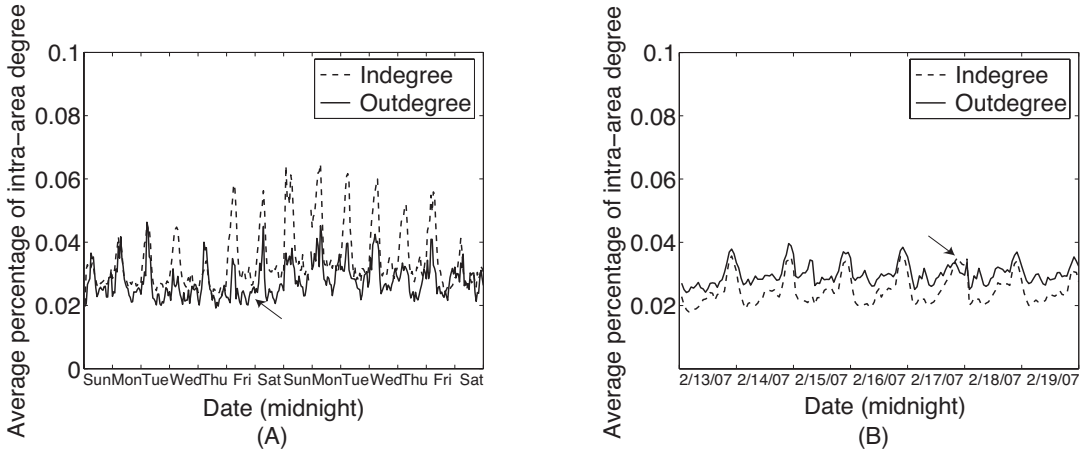


Figure 7.4: Evolution of average intra-area degrees for stable peers in the network. (A) From October 1, 2006 to October 14, 2006. (B) From February 13, 2007 to February 19, 2007.

From the results from both trace periods, we notice that the intra-area degree percentage is very low for both indegree and outdegree (less than 0.062 at all times), implying no area-based clustering in the streaming topology. As link TCP throughput is one major criterion for peer selection in UUSee, for connections inside China, this may also reveal that there does not exist a significant throughput difference between connections in the same province and across different provinces. For peers outside China, they may not have been able to efficiently select peers in nearby regions, which may require further improvements of the UUSee peer selection protocol.

7.1.5 Peer sending throughput vs. outdegree

As the throughput along each P2P connection varies, a peer with a large degree may not necessarily indicate that it is a supernode, *i.e.*, a peer with high throughput. To further explore the relation between peer throughput and degree, we make use of the throughput data along each P2P link as contained in the traces, and compute weighted peer degree

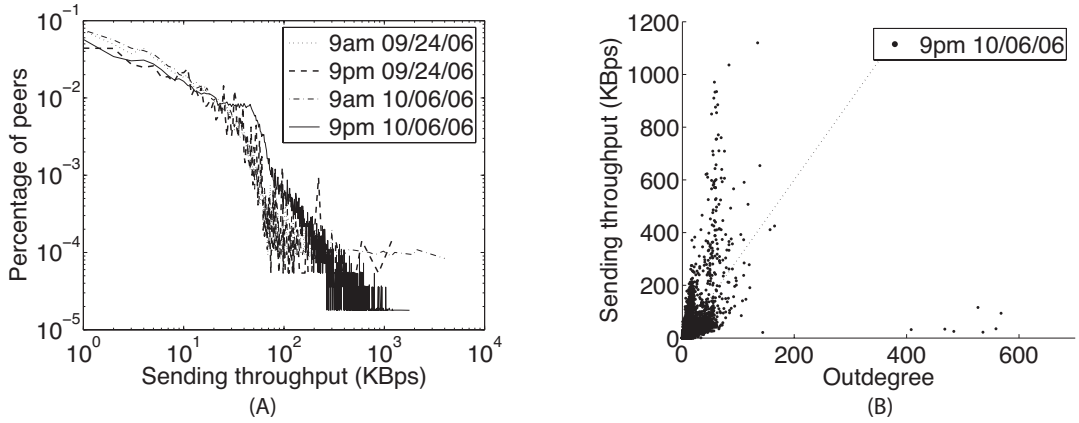


Figure 7.5: (A) Sending throughput distribution of stable peers in the global topology; (B) Correlation between outdegree and sending throughput.

distributions, with link throughput as the link weight. The weighted indegree of a peer at each time is indeed its instantaneous receiving throughput, which is consistently around the streaming rate in such a P2P streaming application. The weighted outdegree of a peer is its instantaneous sending throughput, which is a better indication of the peer's resource availability as a supernode. Therefore, we first investigate the sending throughput distribution across the UUSee overlay, and then examine its correlation with the peer outdegree. The results are given in Fig. 7.5.

Comparing Fig. 7.5(A) and Fig. 7.1(C), we notice that the sending throughput distributions represent similar shapes to corresponding outdegree distributions. Similar to outdegree, sending throughputs in the streaming overlay tend to be larger at peak hours and in the flash crowd scenario. In addition, although occupying a small fraction, there do always exist peers with large upload capacities in the overlay. For example, in the overlay of 9 p.m., October, 6, 2006, there were about 730 peers with higher than 384 KBps aggregate sending throughput, out of 87340 stable peers in the network. We note that we do not include the set of dedicated streaming servers in our topological studies

throughout the chapter, and thus the possibility that any of the discovered supernodes might be a dedicated streaming server is excluded.

While the similarities between Fig. 7.5(A) and Fig. 7.1(C) may imply that a larger outdegree indeed indicates a larger sending throughput in UUSee, we further validate this point by plotting the correlation between the outdegree and sending throughput in Fig. 7.5(B). The plot reveals a positive linear correlation, which is also proven by the calculated Pearson product-moment correlation coefficient between outdegree and sending throughput, at the value of 0.4871.

Finally, we note that such a correlation does not exist between the peer indegree and aggregate receiving throughput, as the receiving throughput is consistently around the streaming rate, while the peer indegree varies significantly, as shown in Fig. 7.1(B).

7.2 Clustering

Studies on the Gnutella network have pointed out that both previous and current generation Gnutella networks exhibit *"small-world"* properties, *i.e.*, peers are highly clustered with small pairwise shortest path lengths, as compared to a random graph of similar peer numbers and link densities. To investigate whether an undirected graph g is a small-world graph, a clustering coefficient is calculated as $C_g = \frac{1}{N}\sum_{i=1}^{N} C_i$, where N is the total number of vertices in the graph, and C_i is the clustering coefficient for vertex i, calculated as the proportion of edges between vertices within its neighborhood to the number of edges that could possibly exist between them [84]. Therefore, a larger clustering coefficient represents more clustering at nodes in the graph. A graph g is identified as a small world if (1) it has a small average pairwise shortest path length l_g, close to that of a corresponding random graph l_r; and (2) a large clustering coefficient C_g, which

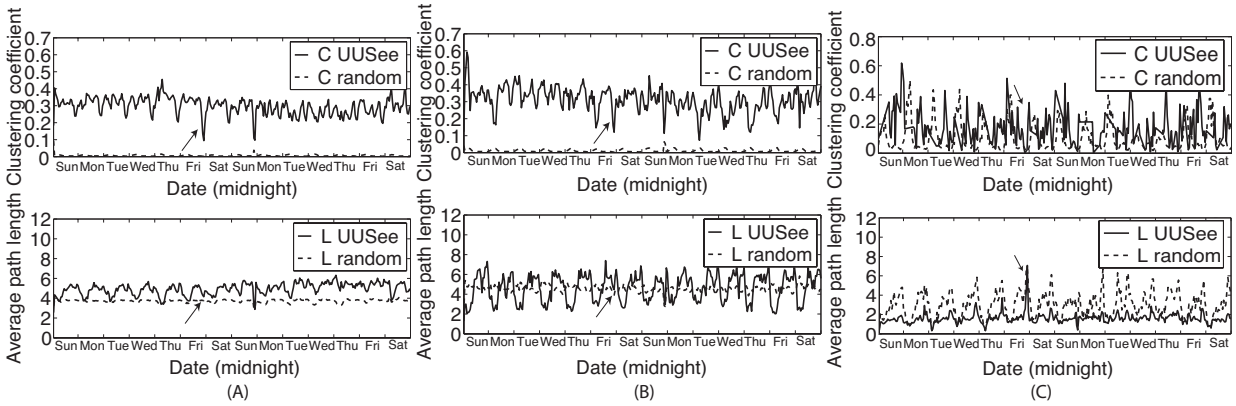


Figure 7.6: Small-world property from October 1, 2006 to October 14, 2006. (A) Small-world metrics for the entire stable-peer graph; (B) Small-world metrics for an ISP subgraph (China Netcom); (C) Small-world metrics for an area subgraph (Zhejiang Province).

is orders of magnitude larger than that of the corresponding random graph C_r.

Based on the traces, we construct a subgraph of the entire UUSee topology at each time, by only including the stable peers and the active links among them. We investigate small-world properties of such stable-peer graphs, and believe they may reveal the connectivity of the original topologies as well.

Fig. 7.6(A) plots the clustering coefficients and average pairwise shortest path lengths of the stable-peer graph over the two-week period. We observe that its clustering coefficients are consistently more than an order of magnitude larger than those of a corresponding random graph, while their average path lengths are similar. This implies that the stable-peer graph does exhibit small-world properties. Besides, we observe slight decreases of clustering coefficients and slight increases of path lengths at peak hours of each day, which may be explained by the broader choice of partners at each peer in larger networks with significantly more peers at the peak times. In Vu *et al.*'s PPLive study, using the same clustering coefficient metric, they show that a less popular channel with

500 nodes is similar to a random graph, while the larger the channel popularity is, the more clustering it becomes. Our study focuses on the entire stable-peer topology, which is composed of tens of thousands of peers at each time, and thus represents the more clustering case discussed in the PPLive study.

Another observation we can make from Fig. 7.6(A) is that, the average pairwise shortest path length is quite steady, consistently around 5 at all times. This implies low network diameters in such stable-peer topologies. Considering that transient peers are connected to one or more stable peers with high probability, we conjecture that the pairwise shortest path lengths in the original UUSee topologies should be close to those in the stable-peer graphs. Therefore, the overall UUSee streaming network may represent a low network diameter, which facilitates the quick distribution of media blocks throughout the entire topology.

In Sec. 7.1.3, we have observed the phenomenon of ISP-based peer clustering. Here, we wish to further validate this finding by calculating the clustering coefficient and average pairwise path length for the subgraph composed of stable peers in the same ISP and active links among them. A representative result is shown in Fig. 7.6(B) with respect to a major China ISP — China Netcom. Comparing Fig. 7.6(B) with Fig. 7.6(A), we have observed that the ISP subgraph has more clustering than the complete topology of stable peers, with (1) closer average path lengths to those of the random graphs, and (2) larger clustering coefficient difference from those of the random graphs. In our study, similar properties were observed for sub-topologies of other ISPs as well.

With the example of the sub streaming topology inside Zhejiang Province in China, we again investigate the clustering coefficient and average pairwise path length over the area sub-topology. Fig. 7.6(C) clearly demonstrates that there is no significant difference

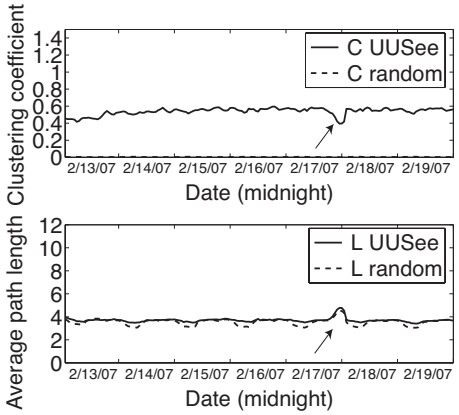


Figure 7.7: Small-world metrics for the entire stable-peer graph: February 13, 2007 to February 19, 2007.

between the magnitudes of clustering coefficients for the area sub-topology and those of the corresponding random networks. Together with similar results from small-world metric evaluations of other geographic areas, it validates that there does not exist area-based clustering in the UUSee streaming network.

To examine whether such small-world properties may be different over a longer period of time, we have further investigated the clustering coefficient and average shortest path length in the entire stable-peer graph using the traces in February, 2007. Comparing the results in Fig. 7.7 to those in Fig. 7.6(A), we are able to identify a higher level of clustering in the UUSee streaming network at the later trace period. Again, decrease of the clustering coefficient and increase of the path length are observed during the flash crowd scenario on February 17, 2007. We omit the results with respect to the existence of ISP-based clustering and non-existence of area-based clustering observed in this period, which are nevertheless similar to those given in Fig. 7.6(B) and (C).

7.3 Reciprocity

In a modern P2P streaming application such as UUSee, a mesh streaming topology is constructed and a BitTorrent-like block distribution protocol is employed over the mesh. However, as all media blocks originate from a collection of dedicated streaming servers and then propagate throughout the network, one may wonder: Is the media content propagating in a tree-like fashion, *i.e.*, a peer retrieves from a set of peers closer to the servers and further serves another group of peers farther away from servers? Or does such mesh-based streaming really benefit from reciprocal media block exchanges between pairs of peers? If it is the latter case, to what extent are the peers reciprocal to each other?

To answer these questions, we investigate another graph property on the P2P streaming topology: *edge reciprocity*. In a directed graph g, an edge (i, j) is reciprocal if vertex j is also linked to vertex i in the reverse direction, *i.e.*, (j, i) is also an edge in the graph. A simple way to obtain reciprocity of a graph is to compute the fraction of bilateral edges over the total number of edges in the graph:

$$r = \frac{\sum_{i \neq j} a_{ij} a_{ji}}{M}, \tag{7.1}$$

where a_{ij}'s are entries of the adjacency matrix of graph g ($a_{ij} = 1$ if an edge exists from i to j, and $a_{ij} = 0$ if not), and M is the total number of edges in the graph. However, this simple reciprocity metric cannot distinguish between networks with high reciprocity and random networks with high link density, which tend to have a large number of reciprocal edges as well, due exclusively to random factors. Therefore, we utilize another more

accurate edge reciprocity metric proposed by Garlaschelli *et al.* [35]:

$$\rho = \frac{r - \bar{a}}{1 - \bar{a}}, \tag{7.2}$$

where r is as defined in (7.1), and $\bar{a}$ is the ratio of existing to possible directed links in the graph, *i.e.*, $\bar{a} = \frac{M}{N(N-1)} = \frac{\sum_{i \neq j} a_{ij}}{N(N-1)}$ with N being the total number of vertices. Since in a random network, the probability of finding a reciprocal link between two connected nodes is equal to the average probability of finding a link between any two nodes, $\bar{a}$ actually represents the reciprocity calculated with (7.1), of a random graph with the same number of vertices and edges as g. Therefore, the edge reciprocity defined in (7.2) is an absolute quantity, in the sense that: if $\rho > 0$, the graph has larger reciprocity than a corresponding random graph, *i.e.*, it is a reciprocal graph; if $\rho < 0$, the network has smaller reciprocity than its random version, *i.e.*, it is an antireciprocal graph.

To compute the reciprocity among all the peers in the UUSee network at one time, we use all the directed active links among peers that appeared in the trace at the time. If streaming in the UUSee network takes place in a tree-like fashion, the computed edge reciprocity should be negative, as its $r = 0$ and $\rho = -\frac{\bar{a}}{1-\bar{a}} < 0$. If there is no strong correlation between the sets of supplying and receiving partners at each peer, the edge reciprocity will be around 0, *i.e.*, the case of a random network. If the peers do help each other materially by exchanging media blocks, the edge reciprocity should be greater than 0, and can be as large as 1.

Fig. 7.8(A) plots the evolution of edge reciprocity in the entire UUSee topology. The consistent greater-than-zero edge reciprocity reveals significant reciprocal exchanges of available blocks among pairs of peers in such mesh-based streaming. It also implies that the sets of supplying and receiving partners at each peer are strongly correlated,

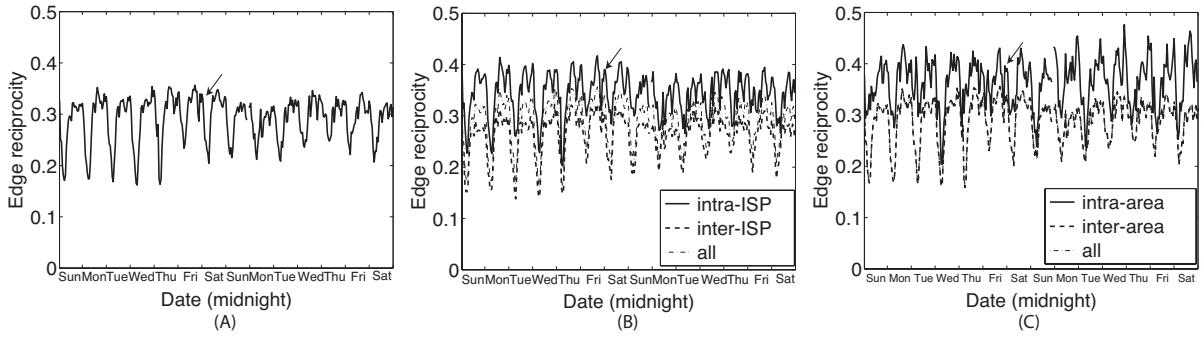


Figure 7.8: Edge reciprocity from October 1, 2006 to October 14, 2006. (A) Edge reciprocity for the entire network; (B) Reciprocity for edges in the same ISP and across different ISPs; (C) Reciprocity for edges in the same area and across different areas.

as compared to a purely random network. Furthermore, the reciprocity exhibits daily evolution patterns with peaks at the peak hours as well.

We have discovered ISP-based clustering of the peers in the previous sections, where the direction of P2P links is not essentially utilized. Now taking connection directions into consideration, we further investigate the reciprocity of links connecting peers in the same ISP and those among peers across different ISPs. For this purpose, we derive two sub-topologies from each topology we used in the previous reciprocity investigation: one contains links among peers in the same ISPs and their incident peers, and the other consists of links across different ISPs and the incident peers. Fig. 7.8(B) shows edge reciprocities for the two sub-topologies. For the purpose of comparison, it also plots the edge reciprocities for the entire topology. We have observed a higher reciprocity for the intra-ISP sub-topology and a lower reciprocity for the inter-ISP sub-topology, as compared to that of the complete topology. This implies that the streaming topology in each ISP is a densely connected cluster with large portions of bilateral links among the peers.

Similarly, we investigate the edge reciprocity for links connecting peers in the same

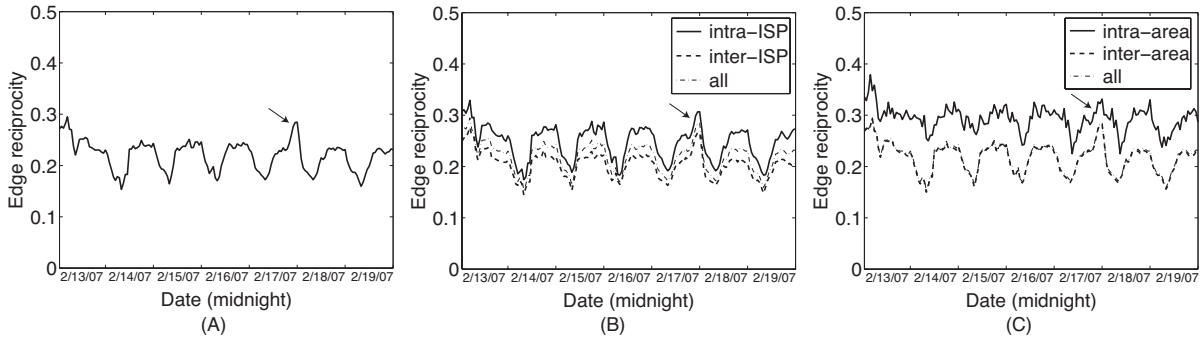


Figure 7.9: Edge reciprocity from February 13, 2007 to February 19, 2007. (A) Edge reciprocity for the entire network; (B) Reciprocity for edges in the same ISP and across different ISPs; (C) Reciprocity for edges in the same area and across different areas.

area and those among peers across different areas, using the intra-area sub-topology and the inter-area sub-topology at each time. While the evolution curve for the inter-area sub-topology mostly overlaps with that of the entire topology, Fig. 7.8(C) reveals high reciprocities in the intra-area sub-topology. This is an interesting discovery, since even though we have observed no clustering for the streaming topology in one area, there does exist a high level of reciprocity over the limited number of intra-area links, *i.e.*, peers in the same area are largely reciprocal to each other.

In addition, the edge reciprocities derived using traces in February, 2007, are given in Fig. 7.9. Comparing Fig. 7.9 to Fig. 7.8, we observe generally smaller edge reciprocities in February, 2007, which we attribute to the expansion of UUSee streaming network over the months, such that each peer has a broader choice of partners. Nevertheless, the other properties, such as the daily evolution pattern and the better reciprocity inside the same ISP or area, remain.

7.4 Supernode Connectivity

In Sec. 7.1.5, we have observed the existence of supernodes in the P2P streaming overlay, *i.e.*, the peers with high sending throughput. Next, we start our investigations on the connectivity among supernodes: Do they tend to connect to each other and constitute a hub in such a practical streaming network? Do they tend to exchange media blocks among each other in a reciprocal way?

To address these questions, we utilize the *likelihood* metric proposed by Li *et al.* in [57], which is also linearly related to the *assortativity coefficient* discussed by Newman [66]. These metrics suggest the connectivity of nodes with similar degrees in a graph, *i.e.*, whether they tend to be tightly interconnected or not. The likelihood metric for undirected graph g is defined as follows by Li *et al.* [57]: Let $L(g)$ be the sum of products of degrees of adjacent nodes, *i.e.*, $L(g) = \sum_{(i,j)\in E(g)} d_i d_j$, where $E(g)$ is the set of edges in graph g and d_i is the degree of vertex i. Let $L_{\max}$ and $L_{\min}$ denote the maximum and minimum values of $L(g)$ among all simple connected graphs with the same number of vertices and the same node degree sequence as graph g. The likelihood is defined as:

$$\text{likelihood}(g) = \frac{L(g) - L_{\min}}{L_{\max} - L_{\min}}. \tag{7.3}$$

In order to compute $L_{\max}$ and $L_{\min}$, we need to first generate graphs that have these likelihood values. A $L_{\max}$ ($L_{\min}$) graph can be generated by the following simple heuristics: Sort nodes in graph g from the highest to the lowest degree. To generate the $L_{\max}$ ($L_{\min}$) graph, connect the highest degree node successively to other high (low) degree nodes in decreasing (increasing) order of their degrees until it satisfies its degree requirement, and then connect the second highest degree node successively to other nodes in decreasing

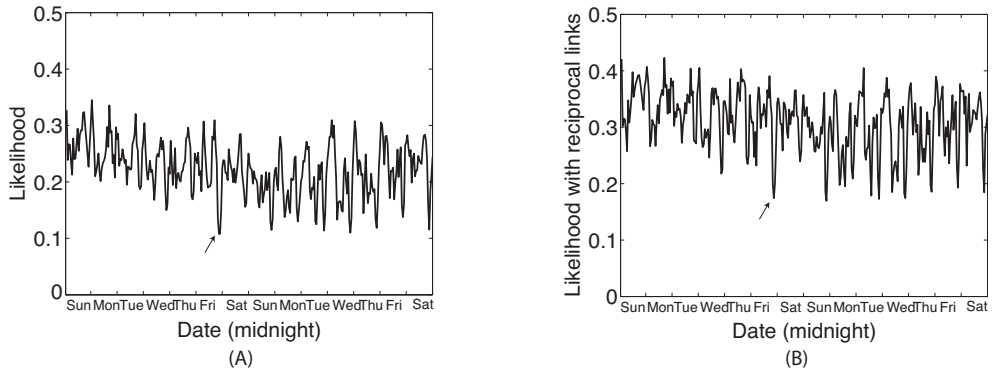


Figure 7.10: Likelihood from October 1, 2006 to October 14, 2006. (A) Likelihood computed with all the links in the UUSee network; (B) Likelihood computed with only reciprocal links in the UUSee network.

(increasing) degree order (which have not saturated their degree requirements) to achieve its degree, etc. This process is repeated for all nodes in descending degree order. In this way, the likelihood computed with (7.3) is a normalized quantity in the range of $[0, 1]$. A close-to-0 likelihood indicates that high degree nodes tend to connect to low degree nodes in the graph, while a close-to-1 likelihood reveals more clustering of the nodes with similar degrees.

To derive the connectivity among supernodes in the UUSee network, instead of using peer degrees in the computation of likelihood, we use the sending throughput of each peer, as we believe it is a better indication of the resource availability of the peers as supernodes. Besides, we have shown in Sec. 7.1.5 that peer sending throughput is positively correlated with outdegree in the UUSee streaming overlay.

We first compute the likelihood using all the active links included in the instantaneous UUSee streaming topologies. Fig. 7.10(A) shows that the likelihood values are below 0.3 at most times. These represent quite low likelihood in its value range of $[0, 1]$, *i.e.*, below the average likelihood among graphs with the same number of nodes and node

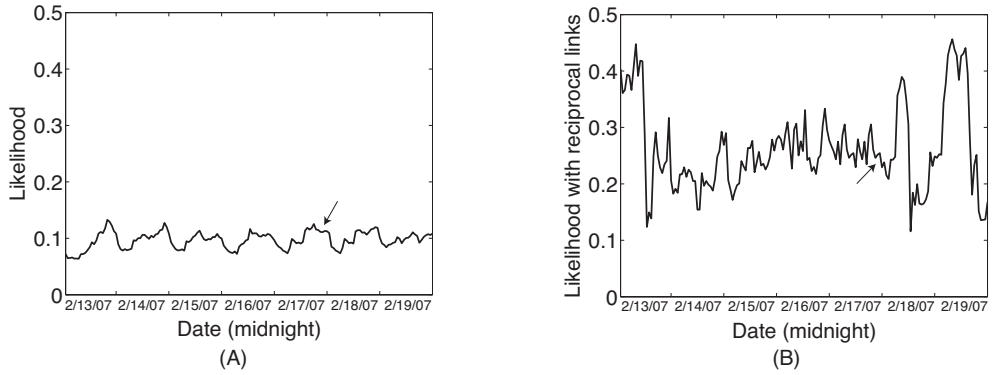


Figure 7.11: Likelihood from February 13, 2007 to February 19, 2007. (A) Likelihood computed with all the links in the UUSee network; (B) Likelihood computed with only reciprocal links in the UUSee network.

sending throughput sequence. This observation indicates that supernodes do not tend to be tightly connected to each other in the UUSee streaming overlay, but are surrounded largely by nodes with lower sending abilities.

Motivated by the reciprocal properties that we have discovered in the previous section, we further explore the following question: Are supernodes more inclined to exchange media content with other peers with comparably high sending throughputs? To answer this question, we compute the likelihood of UUSee streaming network again with only reciprocal links in each instantaneous topology. Comparing Fig. 7.10(B) with Fig. 7.10(A), we find that the likelihood with respect to reciprocal links is generally larger than that in the entire topology, implying that the reciprocal links are relatively more likely to be connecting nodes with comparable sending throughputs. Nevertheless, the likelihood values are still lower than average among the set of graphs with the same number of nodes and the same sequence of node sending throughput.

Investigations of the likelihood property using traces in February, 2007 further validate our above discoveries. As illustrated in Fig. 7.11, the likelihood is at an even lower level

in February, 2007. All these observations exhibit that, in such a practical streaming network, peers are not clustered based on their resource availability, and there does not exist a supernode hub at any given time in the overlay.

7.5 Summary

In this chapter, we present the first extensive effort in the research community to characterize topologies of modern large-scale P2P streaming meshes, based on instantaneous snapshots of the active topology of UUSee. Utilizing a number of meaningful graph metrics, we seek to discover structural properties of the streaming topologies at short time scales, as well as their evolutionary dynamics over longer periods of time. The original insights that we have brought forward in this chapter are the following: (1) The degree distribution towards active neighbors in a P2P mesh does *not* possess similar properties as those obtained from early Internet or AS-level topological studies, such as power-law degree distributions; (2) the streaming topologies naturally evolve into clusters inside each ISP, but not within geographically adjacent areas; (3) peers are reciprocal to each other to a great extent, which contributes to the stable performance of streaming in such mesh networks; (4) there exist a small portion of high-throughput supernodes in the streaming overlay, each assisting a large number of peers with lower bandwidth availabilities, but not tightly connecting to each other in a hub-like fashion.

Chapter 8

Characterizing P2P Streaming Flows

The fundamental advantage of P2P live streaming is to allow peers to contribute their upload bandwidth, such that bandwidth costs may be saved on dedicated streaming servers. It is therefore of pivotal importance for a peer to select other peers with high *inter-peer* bandwidth (*i.e.*, the available bandwidth between two peers) during a live streaming session, in order to retrieve the media content timely to meet the playback deadline. As TCP is widely employed in P2P live streaming applications to guarantee reliability and to transverse NATs, the achievable TCP throughput is an essential metric when evaluating available inter-peer bandwidth.

However, due to the inherent dynamic nature of peer arrivals and departures in a typical P2P streaming session, it is a daunting challenge to evaluate TCP throughput between two peers before data transmission begins. One may start a probing TCP connection to directly measure TCP throughput, but the time it takes for TCP to saturate available bandwidth leads to intrusive and expensive bandwidth usage, that can otherwise be available to stream actual media. A better approach would be to calculate

TCP throughput based on flow sizes, maximum sender/receiver windows, and path characteristics such as delay and loss rate [76, 63, 68]. However, such calculations require the knowledge of TCP parameters or path characteristics, which may not be available without probing or new TCP connections.Yet another alternative is to summarize historical TCP throughput using time series models, which may be utilized to forecast future TCP throughput [39, 59, 82, 85]. Unfortunately, it is common for peers to come across neighbors with whom no historical TCP connections ever exist.

Though it is almost impossible to accurately predict TCP throughput between arbitrary peers without some probing or historical data, practical experiences show that it is *helpful* in the design of a peer selection protocol even if the peer has only a *"rough idea"* about the available bandwidth between itself and a possible candidate, and such a "rough idea" can be used to *rank* the candidates based on available bandwidths. To acquire useful knowledge towards this objective, we represent in this chapter our comprehensive statistical study of TCP throughputs, using 370 million live streaming flows in 230 GB of UUSee traces collected over a four-month period (November 2006 — February 2007).

Our focus in this study is to thoroughly understand and characterize the achievable TCP throughputs of streaming flows among peers in large-scale real-world P2P live streaming sessions, in order to derive useful insights towards the improvement of current peer selection protocol. Using continuous traces over a long period of time, we explore evolutionary properties of inter-peer bandwidth. Focusing on representative snapshots of the entire topology at specific times, we investigate distributions of inter-peer bandwidth in various peer ISP/area/type categories, and statistically test and model the deciding factors that cause the variance of such inter-peer bandwidth.

We have made a number of original discoveries based on our statistical characterization. Our current study has mainly focused on streaming flows within China, but we believe our discoveries also bring useful insights towards global networks. Based on these insights, we design a *throughput expectation index* that facilitates high-bandwidth peer selection without performing any active measurements.

8.1 Throughput Distributions

We start our P2P streaming flow characterization by analyzing the distributions of TCP throughput at representative times, across or within different ISPs/areas, and among different peer types. We note all our flow characterizations in this chapter are based on the 5-minute maximum throughput measurements on the P2P links from the traces, whose collection methodology was elaborated in Sec. 6.2.

8.1.1 Overall throughput distribution at different times

Fig. 8.1 shows the throughput distribution over the entire network at four representative regular times: Monday morning (9am 12/18/06), Monday evening (9pm 12/18/06), Friday morning (9am 12/22/06) and Friday evening (9pm 12/22/06). We note that the bin size used in all our throughput distribution plots in this section is 1KBps. With throughput depicted in the log scale, the plots represent the shapes of normal distributions, corresponding to the original throughputs having the analytic distributions of log-normal [3, 16]. This finding is consistent with existing work of Balakrishnan *et al.* [16] and Zhang *et al.* [89], who also discovered log-normal rate distributions within their Internet flow sets.

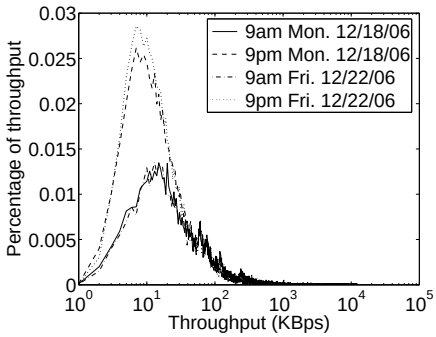


Figure 8.1: Overall throughput distribution at different times.

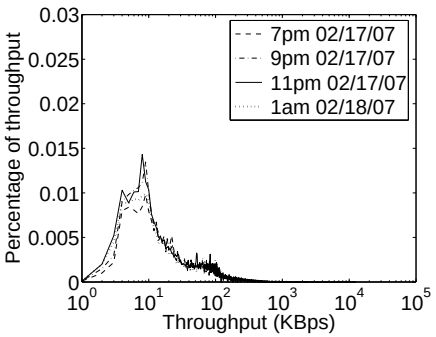


Figure 8.2: Overall throughput distribution at Chinese New Year Eve.

The throughput distributions at the four times peak at 15KBps, 7KBps, 13KBps, 7KBps, respectively, with an 80th percentile of 280KBps, 96KBps, 275KBps, and 90KBps, respectively. We observe that the mean throughputs in the mornings, which are daily off-peak hours for the streaming application, are $2-3$ times higher than those at evening peak hours, and the variance of the throughputs in the mornings is larger than that in the evenings as well. For the same time at different days in a week, however, there does not exist an apparent throughput difference.

We further validate the above observations statistically using one-way analysis of variance (ANOVA) [25, 36]. The one-way ANOVA is used to test the null hypothesis that different sets of samples for an independent variable have all been drawn indifferently from the same underlying distribution. In our case, we use ANOVA to examine whether the throughput distributions at different times on a same regular day are statistically equivalent, and whether those at the same time on different days are significantly different. As the numbers of throughput samples in the four sets are different, we conduct ANOVA by using the non-parametric Kruskal-Wallis test [36]. The comparisons and reported p-values are listed in Table 8.1.

In our hypothesis test, if a result p-value is lower than the significance level of 0.05,

Table 8.1: Kruskal-Wallis ANOVA for throughputs at different times

Null Hypothesis	Throughput Sets	p-value
The two sets of	9am Mon. vs. 9am Fri.	0.8699
throughputs have	9pm Mon. vs. 9pm Fri.	0.0684
the same distribution	9am Mon. vs. 9pm Mon.	0
	9am Fri. vs. 9pm Fri.	0

the difference between the corresponding distributions is statistically significant, and the null hypothesis is rejected; otherwise there is insufficient evidence to reject the null hypothesis. The 0 p-values reported for the latter two tests strongly suggest the difference between throughputs at different times of a day, while the other large p-values validate the large similarity among morning/evening throughput sets on different days.

While the above observations generally apply for throughput sets on regular days, we have also investigated throughput distributions during a flash crowd scenario on Chinese New Year Eve (Feb. 17th, 2007), as shown in Fig. 8.2. Four representative snapshots are plotted: 7pm on the Eve, before the celebration TV broadcast started; 9pm, when the flash crowd started to gather as more and more viewers tuned in to the channel; 11pm, when the flash crowd reached its largest size as the Chinese New Year approached; and 1am on the next morning, when the crowd dismissed itself after the show ended. With ANOVA tests, we detected that the distributions are statistically different, with throughputs at 7pm statistically larger than those at 1am, followed by those around 9pm, and then those at 11pm. This reflects that inter-peer bandwidths became tight as the size of flash crowd increased and turned loose again when the crowd dismissed. Nevertheless, there does not exist a "crash" scenario with abrupt drop of throughput over the network, and the throughputs follow similar log-normal distributions as those at the same time on a regular day.

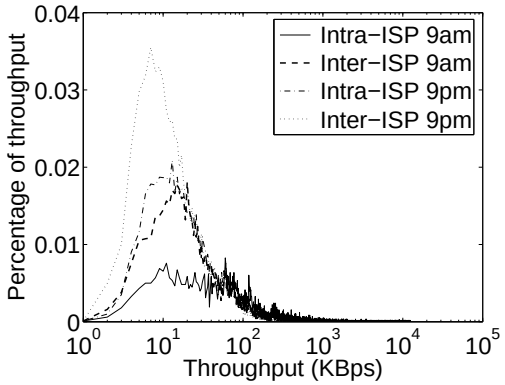


Figure 8.3: Intra/inter ISP throughput distribution on Dec. 18, 2006.

8.1.2 Intra/inter ISP throughput distribution

Using the IP-to-ISP/area mapping database described in Sec. 6.3.3, we next categorize the P2P streaming flows into two classes and investigate their respective throughput distributions: (1) intra-ISP flows, for which the sender and receiver are in the same ISP, and (2) inter-ISP flows, where they belong to different ISPs. Fig. 8.3 exhibits that, while they still follow log-normal distributions in each category, intra-ISP throughputs are generally larger than their inter-ISP counterparts measured at the same time: the former have peaks at 60KBps (9am) and 13KBps (9pm), while those of the latter are 20KBps (9am) and 7KBps (9pm), respectively. Also observed is that intra-ISP throughputs at peak hours are generally smaller than inter-ISP throughputs at off-peak hours on the same day. Within each intra-ISP or inter-ISP category, the throughput distributions show a similar diurnal pattern as that revealed by the overall throughput distributions in the previous section: both the mean and variance of the throughput distributions in the mornings are larger than those in the evenings.

While these observations meet our general expectation that bandwidth is more abundant within each ISP, we also notice many large inter-ISP throughput values and the

large span for both inter-ISP and intra-ISP throughputs. This inspires us to further investigate: Are throughputs for flows within an ISP always statistically larger than those for flows to and from this ISP? Is there significant throughput difference across different pairs of ISPs? To answer these questions, we again conduct Kruskal-Wallis ANOVA tests to various throughput sets categorized based on contingent ISPs of the flows. If 3 or more throughput sets are compared in one test and significant difference is reported, we further perform the *multiple comparison test (or procedure)* [25, 36] to investigate the difference between each pair of sets. The representative tests and their results are given in Table 8.2. To conserve space, we use the following abbreviations for ISPs: TC (Telecom), NC (Netcom), UC (Unicom), TT (Tietong), Edu (Education Network).

Again, taking 0.05 as the p-value threshold to determine if we should reject the null hypothesis, our discoveries from the ANOVA are the following. *First*, inter-ISP throughputs are not necessarily smaller than their intra-ISP counterparts. For the two largest China ISPs, Netcom and Telecom, the throughputs of their inbound flows are generally smaller than those of their internal flows. Throughputs are especially small between the two ISPs themselves. For every other ISP, there is no significant throughput difference among the internal flows and inbound flows. This validates the fact that there is a stringent bandwidth constraint between Netcom and Telecom, as two major ISP competitors in China, while no such caps exist across the other small ISPs and between those two and the small ISPs. *Second*, throughput asymmetry is exhibited from one direction to the other across the two largest ISPs, as well as between them and the other ISPs. The observation that throughput from large ISPs to small ISPs are smaller than those in the other direction may reveal possible bandwidth caps placed by large ISPs on such relay traffic.

Table 8.2: Kruskal-Wallis ANOVA for throughputs across different ISPs at 9pm, Dec. 18, 2006

Null Hypothesis	Throughput Sets	p-value	Multiple Comparison Test Result
Throughput sets within an ISP and from different ISPs to this ISP have the same distribution	(1) TC→TC, NC→TC, UC→TC, TT→TC, EDU→TC	0	$\text{Throughput}_{TC\to TC} \approx \text{Throughput}_{UC\to TC} \approx \text{Throughput}_{EDU\to TC} > \text{Throughput}_{TT\to TC} > \text{Throughput}_{NC\to TC}$
	(2) NC→NC, TC→NC, UC→NC, TT→NC, EDU→NC	0	$\text{Throughput}_{NC\to NC} \approx \text{Throughput}_{UC\to NC} \approx \text{Throughput}_{TT\to NC} > \text{Throughput}_{EDU\to NC} \approx \text{Throughput}_{TC\to NC}$
	(3) UC→UC, TC→UC, NC→UC, TT→UC, EDU→UC	0.062	
	(4) TT→TT, TC→TT, NC→TT, UC→TT, EDU→TT	0.081	
Throughput set from ISP1 to ISP2 and throughput set from ISP2 to ISP1 have the same distribution	(1) TC→NC, NC→TC	0.032	$\text{Throughput}_{TC\to NC} > \text{Throughput}_{NC\to TC}$
	(2) TC→UC, UC→TC	0.023	$\text{Throughput}_{UC\to TC} > \text{Throughput}_{TC\to UC}$
	(3) NC→TT, TT→NC	0.029	$\text{Throughput}_{TT\to NC} > \text{Throughput}_{NC\to TT}$
	(4) UC→TT, TT→UC	0.396	
	(5) EDU→UC, UC→EDU	0.153	

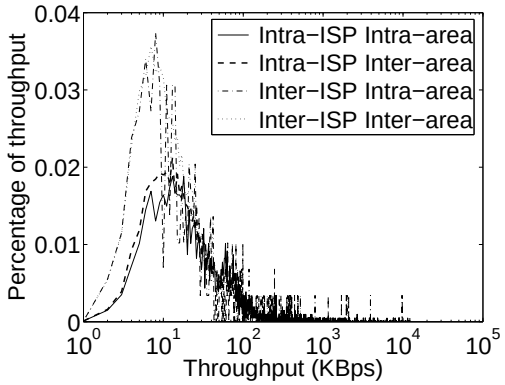


Figure 8.4: Intra/inter area throughput distribution at 9pm, Dec. 18, 2006.

8.1.3 Intra/inter area throughput distribution

To characterize the P2P streaming flow at finer granularity below the ISP level, we next compare throughput distributions in the cases that the sender and receiver are located within or not within the same geographic area (intra-area vs. inter-area). Here, the peers are in the same area if they are in the same province of China. As we have concluded that ISP memberships of the peers may significantly affect the inter-peer bandwidth, we investigate four cases, as shown in Fig. 8.4. When ISP memberships are fixed, we observe no significant difference between the distributions of intra-area throughputs and inter-area throughputs; in either area case, intra-ISP throughputs are always larger than inter-ISP throughputs. To validate these observations, we again perform ANOVA to test the difference between the intra-area throughput set and inter-area throughput set for each specific ISP pair, and the difference among throughput sets from different ISPs to one ISP in both the intra-area and inter-area cases. The representative tests and their results are given in Table 8.3.

Comparing the p-values with threshold 0.05, we first find that, in both cases that the sender and receiver do and do not belong to the same ISP, there does not exist a

Table 8.3: Kruskal-Wallis ANOVA for inter/intra area throughputs between different ISPs at 9pm, Dec. 18, 2006

Null Hypothesis	Throughput Sets	p-value
Inside the same ISP, intra-area throughput set and inter-area throughput set have the same distribution	(1) intra-TC: intra-area set v.s. inter-area set	0.2396
	(2) intra-NC: intra-area set v.s. inter-area set	0.0701
	(3) intra-TT: intra-area set v.s. inter-area set	0.6228
	(4) intra-UC: intra-area set v.s. inter-area set	0.5751
Across two different ISPs, intra-area throughput set and inter-area throughput set have the same distribution	(1) TC→NC: intra-area set v.s. inter-area set	0.117
	(2) NC→TC: intra-area set v.s. inter-area set	0.179
	(3) NC→TT: intra-area set v.s. inter-area set	0.3105
	(4) UC→TT: intra-area set v.s. inter-area set	0.4575
Inside the same area, throughput sets within one ISP and from different ISPs to this ISP have the same distribution	(1) TC→TC, NC→TC, UC→TC, TT→TC, EDU→TC	0.0015
	(2) NC→NC, TC→NC, UC→NC, TT→NC, EDU→NC	0.0448
	(3) UC→UC, TC→UC, NC→UC, TT→UC, EDU→UC	0.5846
	(4) TT→TT, TC→TT, NC→TT, UC→TT, EDU→TT	0.5511
Across two different areas, throughput sets within one ISP and from different ISPs to this ISP have the same distribution	(1) TC→TC, NC→TC, UC→TC, TT→TC, EDU→TC	0
	(2) NC→NC, TC→NC, UC→NC, TT→NC, EDU→NC	0
	(3) UC→UC, TC→UC, NC→UC, TT→UC, EDU→UC	0.052
	(4) TT→TT, TC→TT, NC→TT, UC→TT, EDU→TT	0.2929

significant throughput difference when the sender and receiver are further in or not in the same area (province). For the two nation-wide ISPs, Telecom and Netcom, considering the fact that they are organized on the provincial basis, our discovery shows that within each of them, the inter-province bandwidth constraints do not have apparent negative impact on inter-province P2P flow throughputs. In addition, across the two ISPs, a same provincial locality of two peers does not help in improving the inter-peer bandwidth. This can be explained by the facts that the two ISPs have only 4-6 fixed peering points across China, and even if two peers are in the same province, the underlying links in between them may well go via a peering point that is thousands of kilometers away. Second, in both cases that the sender and receiver are in and not in the same area (province), the comparisons of throughputs from different ISPs (including itself) to the same ISP exhibit similar results as those we have shown in Table 8.2. While area information is included in the comparisons in Table 8.3 but not in Table 8.2, they both show that, for large ISPs, there exist differences between their internal throughputs and those from other ISPs to them; for small ISPs, no difference is identified among the different throughput sets.

All these results lead to the conclusion that ISP membership has more significant impact on inter-peer bandwidths, as compared to geographic locations. In what follows, we mainly focus on ISP memberships when we discuss deciding factors that affect bandwidth availability in the middle of a P2P link.

8.1.4 Throughput distribution for different peer types

To discover the impact of peer types (*i.e.*, peer last-mile bandwidths) on inter-peer bandwidth, we further categorize intra-ISP and inter-ISP flows based on types of their incident

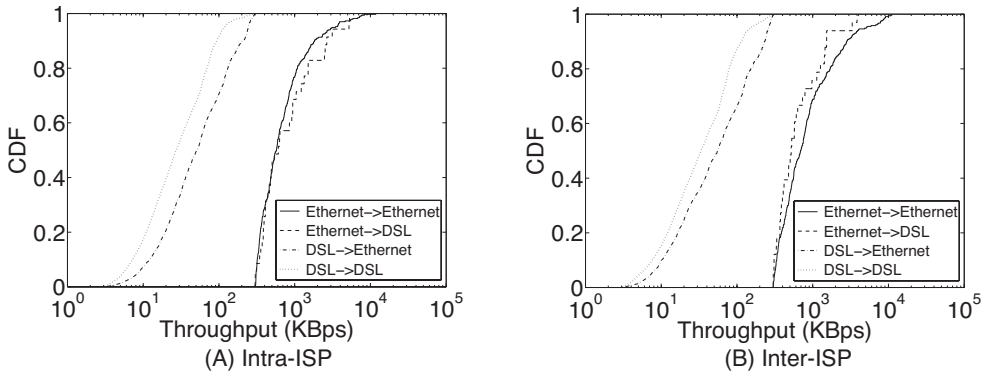


Figure 8.5: Throughput CDF for different peer types at 9pm, Dec. 18, 2006.

peers, and plot the CDF of throughput in each category in Fig. 8.5. The plots and accompanying ANOVA tests exhibit that: throughputs are significantly larger when the sender is an Ethernet peer, in both the intra-ISP and inter-ISP cases; for the same sender type, flows with Ethernet receivers achieve higher throughput in most cases.

The above results reveal a coarse positive correlation between inter-peer bandwidth and the last-mile bandwidths at the peers. It inspires us to further consider the following questions: Is the peer last-mile download/upload capacity the key factor that decides inter-peer bandwidth, both when the peers are in the same ISP and when they are across any pair of ISPs? Or is inter-ISP peering the most important factor that affects throughput between some ISPs? In the following section, we seek to answer these questions with regression modeling of the throughputs.

8.2 Throughput Regression: Focusing on One Snapshot

Focusing on one representative regular snapshot of the UUSee streaming network at 9pm December 18 2006, we investigate the impact of the following factors on inter-peer bandwidths: (1) ISP memberships of the peers, and (2) end-host characteristics, including upload/download capacities and the number of contingent sending/receiving TCP connections at the sender/receiver. We divide our discussions into two cases, intra-ISP case and inter-ISP case, and perform regression analysis on TCP throughputs and the respective end-host characteristics in each case.

8.2.1 Intra-ISP throughput regression

With the example of China Netcom, we check the correlation between flow throughputs on its internal P2P links and various end-host characteristics at the peers. To eliminate outliers and better capture the correlation, we divide the values of each capacity factor into small bins with a width of 5KBps, and plot the median throughput of flows falling into each bin at different levels of capacities in Fig. 8.6. The calculated Pearson product-moment correlation coefficient between throughput and a respective factor, *rho*, is marked at the upper right corner in each plot.

Fig. 8.6(A) exhibits that no significant linear correlation exists between throughput and the upload capacity at the sender peer, especially when the latter is large, *i.e.*, the Ethernet peer case. On the other hand, throughput and download capacity at the receiver peer are better correlated, as shown in Fig. 8.6(B). Nevertheless, there exist many cases in which the throughput is small when the capacity is large. Such unsatisfactory

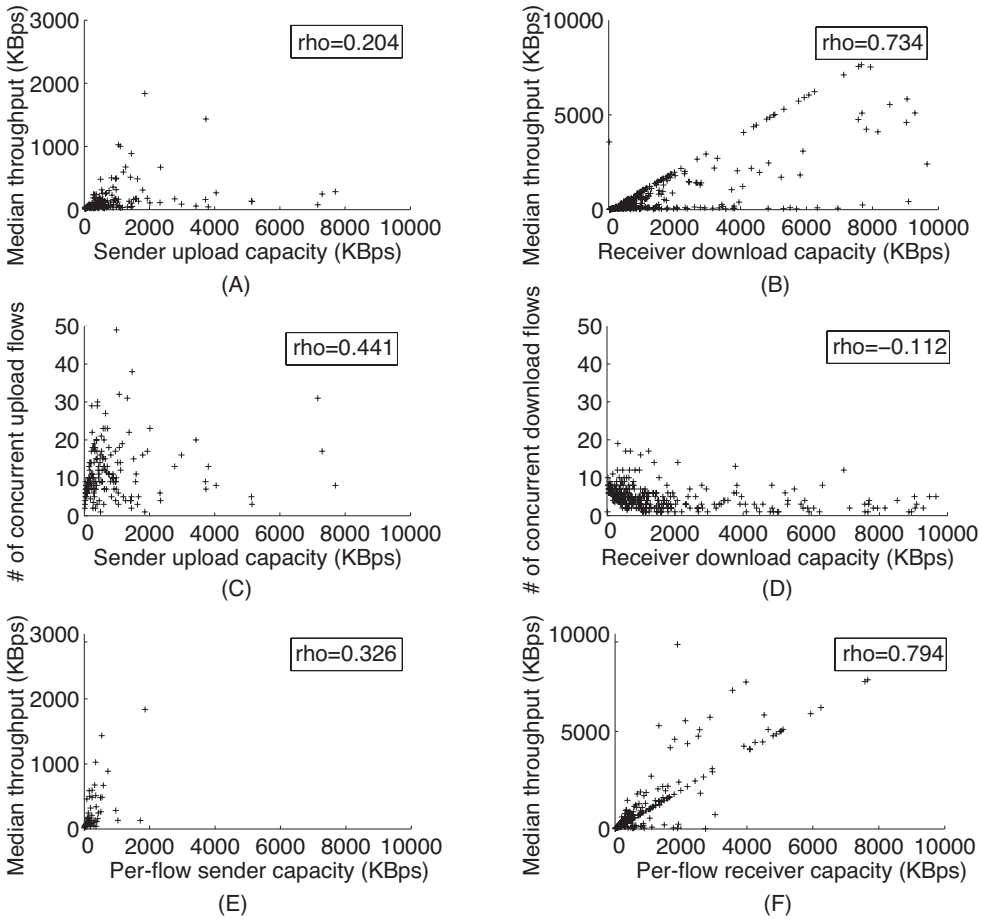


Figure 8.6: Correlation of throughput with end-host characteristics for intra-Netcom flows at 9pm, Dec. 18, 2006.

correlations inspire us to consider: Is the number of contingent upload/download flows high when the upload/download capacity is large, such that the bandwidth share for each flow is small? To answer this question, Fig. 8.6(C) shows a positive correlation between the upload capacity at the senders and the number of their concurrent upload flows, while no significant correlation is exhibited between receiver download capacities and their numbers of concurrent download flows in Fig. 8.6(D). The positive correlation in the former case can be explained by the UUSee streaming protocol design, which

maximally utilizes upload capacity at each peer to serve more neighbors.

Naturally, we then investigate the correlation between throughput and per-flow upload/download bandwidth availability at the sender/receiver, defined as:

$$\textit{per-flow sender capacity} = \frac{\textit{sender upload capacity}}{\textit{no. of concurrent upload flows}},$$
$$\textit{per-flow receiver capacity} = \frac{\textit{receiver download capacity}}{\textit{no. of concurrent download flows}}.$$

Fig. 8.6(E) and (F) exhibit that these two characteristics constitute better explanatory variables towards the throughput regression.

When we take the minimum of per-flow sender capacity and per-flow receiver capacity, we obtain the best deciding factor of throughput, referred to as *per-flow end capacity (PEC)*:

$$PEC = \min(\textit{per-flow sender capacity, per-flow receiver capacity}).$$

Its excellent positive correlation with throughput, with a correlation coefficient of 0.81, is plotted in Fig. 8.7. We next fit PEC and throughput into a linear regression model:

$$\textit{Throughput} = \beta_0 + \beta_1 \times PEC + \epsilon, \tag{8.1}$$

where PEC is the explanatory variable, *Throughput* is the response variable, y-intercept β_0 and slope β_1 are regression coefficients to be estimated, and ϵ denotes the error term.

The basic assumption for least-squares based linear regression analysis is that the response variable is normally distributed. However, as we have shown in Sec. 8.1, throughputs follow approximate log-normal distributions, in which the few large tail values tend to have a strong influence on the regression model. Therefore, we employ *robust linear*

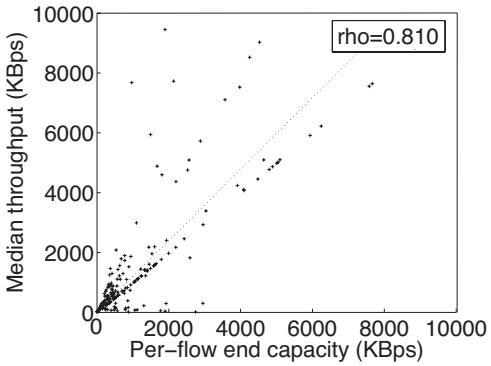


Figure 8.7: Correlation of throughput with per-flow end capacity for intra-Netcom flows at 9pm, Dec. 18, 2006.

regression [65, 72], which uses an iteratively re-weighted least-squares algorithm and is less sensitive to outliers by giving them lower weights. The derived regression statistics are given in Table 8.4.

Table 8.4: Robust linear regression statistics for intra-Netcom throughputs at 9pm, Dec. 18, 2006

β_0 (y-intercept)	β_1 (slope)	p-value for testing significance of β_0	p-value for testing significance of β_1
20.4228	1.1499	0	0

The two p-values are results from tests of the following two null hypotheses, respectively: (1) the y-intercept is "0" (*i.e.*, the y-intercept is non-significant), and (2) the slope is "0" (*i.e.*, the slope is non-significant). As a 0 p-value rejects a corresponding null hypothesis and confirms the significance of regression, the statistics in Table 8.4 further establish the linear correlation between PEC and throughput on intra-ISP flows. In addition, theoretically we expect the regression line to pass through the origin and the slope to be approximately at 45°, and these are validated by the small y-intercept value (as compared to peer last-mile capacities) and near-1 slope value.

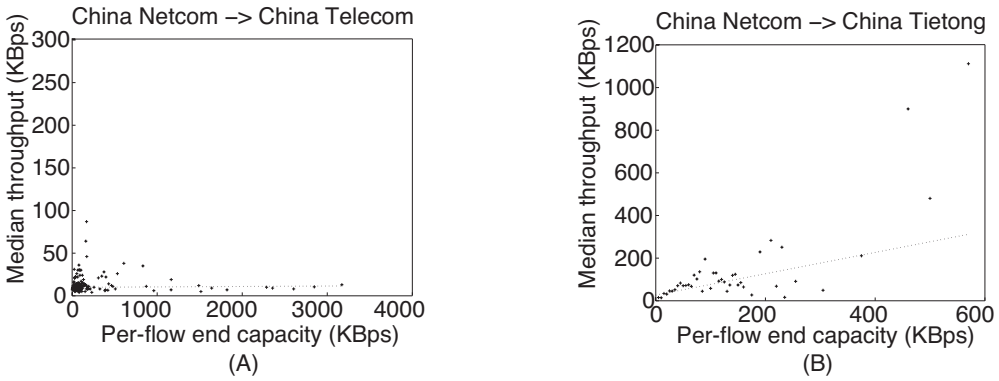


Figure 8.8: Correlation of throughput with per-flow end capacity for inter-ISP flows at 9pm, Dec. 18, 2006.

We have conducted the same regression analysis to throughputs within other ISPs and observed similar correlations. Therefore, we may conclude that, within each ISP, inter-peer bandwidth bottleneck mainly lies at the end hosts, decided by peer last-mile capacities and their concurrent upload/download load: if the capacity bottleneck occurs at the upstream peer, *i.e.*, $\frac{sender\ upload\ capacity}{no.\ of\ concurrent\ upload\ flows} < \frac{receiver\ download\ capacity}{no.\ of\ concurrent\ download\ flows}$, the throughput is limited by the per-flow sender capacity; otherwise, the throughput is decided by the per-flow receiver capacity.

8.2.2 Inter-ISP throughput regression

When it comes to the inter-ISP case, we are interested to explore whether per-flow end capacity still poses a significant impact on inter-peer bandwidth, or it is shadowed by the inter-ISP peering bandwidth bottlenecks. We find the answer is different towards different ISP pairs.

Fig. 8.8(A) exhibits that no significant correlation exists between PEC and throughput for flows from Netcom to Telecom. This is further confirmed by its robust regression analysis statistics in Table 8.5: a p-value of 0.6932 reveals the non-significance of the slope

at the value of 0.0005. Nevertheless, when we investigate streaming flows from Netcom to Tietong, Fig. 8.8(B) shows a different result: throughput is linearly correlated with PEC with a slope of 0.4355, and a corresponding p-value of 0 indicates its significance. The regression statistics for representative flow groups between other ISPs are also listed in Table 8.5.

Table 8.5: Robust linear regression statistics for inter-ISP throughputs at 9pm, Dec. 18, 2006

Throughput Set	β_0 (y-intercept)	β_1 (slope)	p-value for testing significance of β_0	p-value for testing significance of β_1
NC→TC	9.9526	0.0005	0	0.6932
TC→NC	20.4998	−0.0023	0	0.6585
NC→TT	30.5784	0.4355	0	0
TT→NC	39.094	0.316	0	0
TC→TT	24.1774	0.1109	0	0.0480
TT→TC	27.3144	0.5421	0	0
UC→Edu	20.2793	0.7098	0	0
Edu→UC	25.0535	0.4576	0	0

The statistics in Table 8.5 exhibit that: between the two largest ISPs, Netcom and Telecom, throughput is not contingent upon PEC, but limited by their peering bandwidth bottleneck; across other small ISPs and between other ISPs and the two, flow throughput is more or less decided by the peer last-mile bandwidth availability. In addition, in the latter cases, the regression slopes are generally smaller than those obtained for intra-ISP flows, revealing impact of inter-ISP peering. The unexpected discovery, that no apparent bandwidth limitations exist between the large and small ISPs, is quite interesting, especially if we consider the fact that large ISPs levy expensive bandwidth charges on small ones for relaying their traffic in both directions. This may be explained by that small

regional ISPs have to rely on the large nation-wide ISPs to deliver both their outbound and inbound traffic to and from the Internet.

Before we conclude this section, we add that besides regression study on the above snapshot on a regular day, we have also conducted regression analysis on snapshots during the Chinese New Year flash crowd scenario, and have observed similar correlations.

8.3 Throughput Evolution: Time Series Characterization

With the one-time regression model derived, we now switch our focus to evolutionary characteristics of inter-peer bandwidth over time. Such an evolution of bandwidth is due to (1) the variation of the number of concurrent upload/download flows at the sender/receiver; and (2) the dynamics of cross traffic over the P2P links. Here, we are more concerned about the inter-peer bandwidth evolution caused by the latter. Based on the regression model we summarized in Sec. 8.2, we are able to separate effects of the two causes, as the variation of coefficients in the linear models reflects the evolution of bandwidth availability over the P2P links when the per-flow end capacity is kept the same.

8.3.1 Intra-ISP throughput evolution

We now inspect the evolution of bandwidth availability over the internal P2P links of each ISP, by first studying the evolution of coefficients in the linear models, summarized with each of the continuous-time snapshots. Fig. 8.9 plots the evolution of regression coefficients during the week of December 17 — 23, 2006, again with the example of

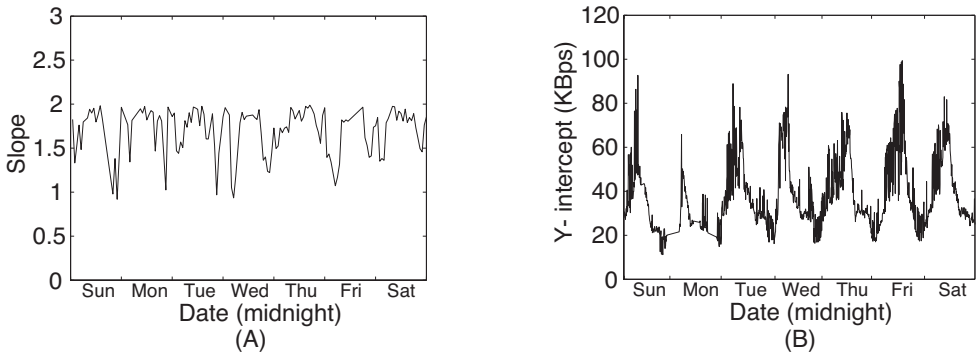


Figure 8.9: Evolution of regression coefficients for intra-Netcom flows in the week of Dec. 17 — 23, 2006.

intra-Netcom flows.

Fig. 8.9 exhibits an apparent daily evolutionary pattern for y-intercept, whose value gradually increases at early hours of a day, peaks around $9-10$ am, and then drops and reaches the lowest around $10-11$ pm. The value of slope, although not as apparent, also shows a similar evolutionary pattern. Not shown in the figures is that p-values for testing the significance of slopes and y-intercepts are always asymptotically zero, exhibiting the significance of throughput regression for inter-Netcom flows at any time.

Though illustrated with the representative week only, the daily evolutionary pattern of regression coefficients — thus bandwidth availability on intra-Netcom P2P links — generally exists during the entire period of the traces. To validate this, we plot in Fig. 8.10 the evolution of mean throughput of intra-Netcom flows over more than 10 weeks of time, from December 10, 2006 to February 21, 2007. To eliminate the effect of varying numbers of concurrent flows at the peers, the mean throughput at each time is calculated as the average of those flow throughputs with PEC in the range of $50-100$ KBps at that time. We observe a daily evolutionary pattern throughout the period, although it is more apparent on some days than others. Daily pattern aside, we also observe a few

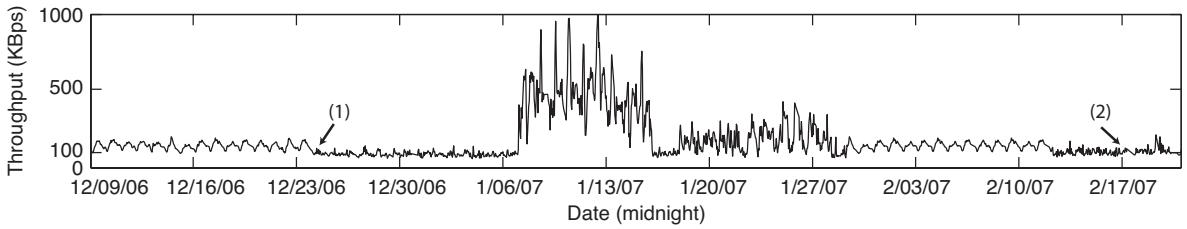


Figure 8.10: Mean throughput evolution for intra-Netcom flows: (1) Taiwan earthquake, (2) Chinese New Year Eve.

abrupt changes of the mean throughput *level* during this period: one around December 26, 2006, the date when an earthquake occurred in the Strait southwest of Taiwan, and another around January 8th, 2007. As the Taiwan earthquake damaged several undersea cables and disrupted some major overseas network connections of China, we conjecture that the first abrupt downgrade of bandwidth is caused by re-routing of traffic, which was originally directed towards overseas servers, to local servers, and the resulting tightness of bandwidths on local connections. We are not quite sure about the reason for the increase of throughput level around mid-January, while we conjecture that it might be caused by ISP upgrades, or measures taken by the ISP to counter the impact of the earlier earthquake. In addition, during the flash crowd scenario on Chinese New Year Eve, we observe no significant throughput downgrade.

Similar observations have been made during investigations of throughput evolution inside other ISPs. All these reveal that although the level of mean throughput may shift, the bandwidth availability on internal P2P links of an ISP statistically evolves following a daily pattern, which persists throughout the trace period.

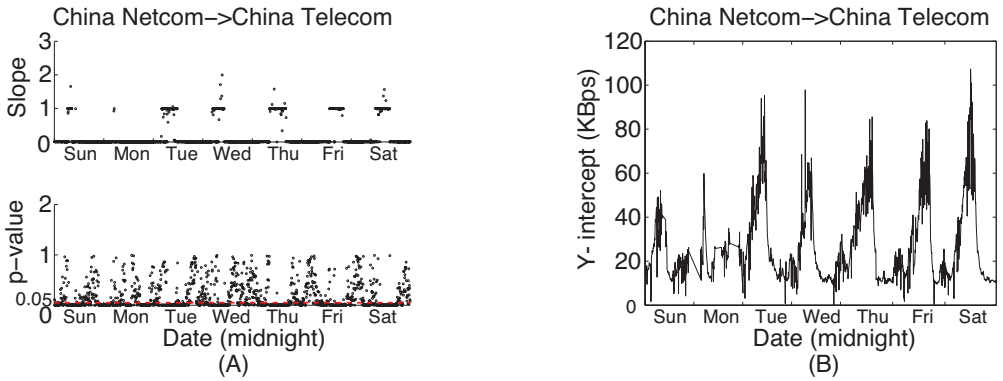


Figure 8.11: Evolution of regression coefficients for Netcom→Telecom flows in the week of Dec. 17 — 23, 2006.

8.3.2 Inter-ISP throughput evolution

In the inter-ISP case, we seek to answer the following questions: First, between Netcom and Telecom, does their inter-ISP peering always limit their inter-ISP P2P flow throughput? If so, is the bottleneck bandwidth availability varying at different times? Second, between the other ISP pairs, how does the bandwidth availability evolve over their inter-ISP links, and does per-flow end capacity always critically decide the throughput at any time?

With the example of the representative week, Fig. 8.11(A) reveals that, at most times of a day between Netcom and Telecom, P2P flow throughput is capped and is not correlated with per-flow end capacity, with a slope of approximately 0 and a corresponding p-value above 0.05. However, there does exist a certain period of time each day when throughputs are significantly correlated with PEC, usually in the early mornings, with slope values around 1 and corresponding p-values below 0.05. In addition, Fig. 8.11(B) exhibits daily evolutionary pattern for the y-intercept, which peaks at the time when the slope is well above 0 on a daily basis.

Based on the estimation algorithm of regression coefficients, we note that when the

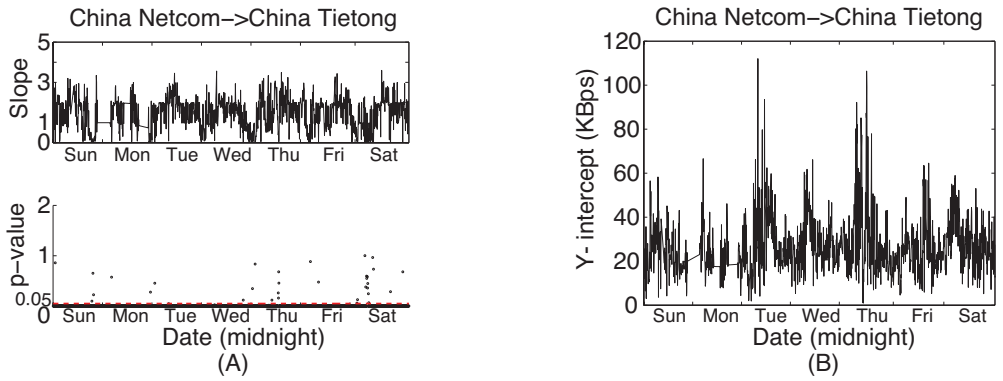


Figure 8.12: Evolution of regression coefficients for Netcom→Tietong flows in the week of Dec. 17 — 23, 2006.

slope is non-significant, the y-intercept represents the mean throughput of the flows between the two ISPs; when the slope is significant, the throughput is decided by peer last-mile bandwidth availability, and does not show apparent inter-ISP peering caps. Therefore, the above observations reveal that: between the two largest ISPs, the limited bandwidth availability gradually improves at early times of a day, peaks in the morning when peer last-mile bandwidths come into play to decide the throughput, then drops and represents the lowest values in the evening.

Next, we inspect the throughput evolution between large ISPs and small ISPs. Fig. 8.12(A) exhibits that, for most of the time between Netcom and Tietong, the inter-ISP throughputs are significantly correlated with PEC, with non-zero slopes and near-zero p-values. Only occasionally at certain moments, there are observed drops of bandwidth availability, when the inter-ISP throughputs are limited regardless of the peer last-mile bandwidth availability. There also exists a daily evolutionary pattern for both the slope and y-intercept, similar to those in the previous cases, although not as apparent.

To validate the above observations in a longer period of time, we again plot the mean throughput evolution for Netcom→Telecom flows with PEC in the range of $10-60$

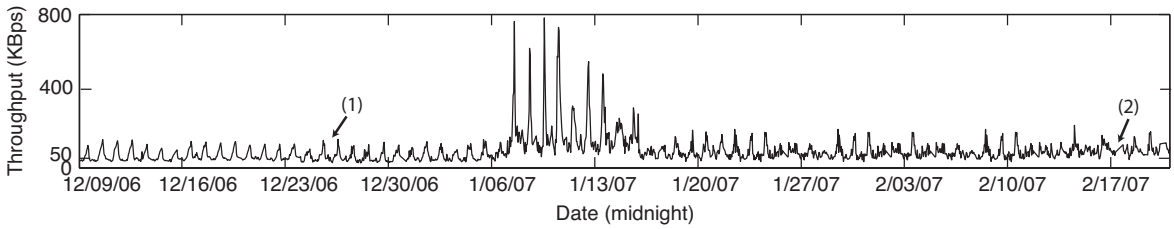


Figure 8.13: Mean throughput evolution for Netcom→Telecom flows: (1) Taiwan earthquake, (2) Chinese New Year Eve.

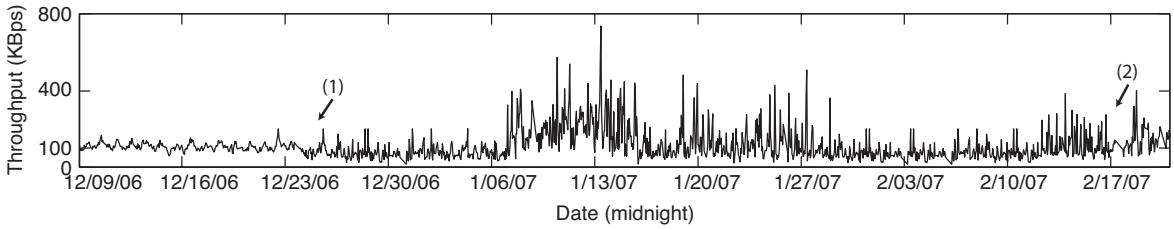


Figure 8.14: Mean throughput evolution for Netcom→Tietong flows: (1) Taiwan earthquake, (2) Chinese New Year Eve.

KBps in Fig. 8.13, and that for Netcom→Tietong flows with PEC in the range of $50-100$ KBps in Fig. 8.14. We also observe the rise of throughput levels in mid-January, but no apparent bandwidth downgrades around the earthquake scenario or flash crowd scenario on Chinese New Year Eve. Nevertheless, the daily evolutionary pattern of throughput persists at all times.

Similar observations have been made in investigations for other ISP pairs. Besides the daily throughput pattern, these observations also reflect that, no apparent inter-ISP bandwidth bottlenecks exist between a large ISP and a small one, and across small ISPs at most times. This again confirms that small ISPs do not usually impose low bandwidth caps at their peering point with large ISPs, in order to facilitate their traffic in both directions.

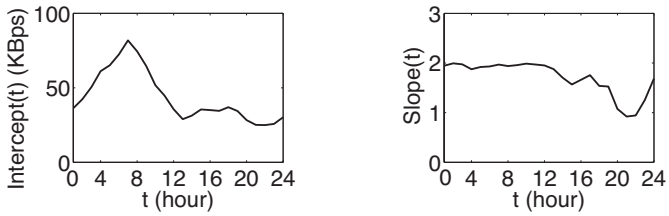


Figure 8.15: Daily intercept/slope functions for intra-Telecom flows.

8.4 Throughput Expectation Index

The throughput characteristics we have derived in previous sections bring useful insights towards the improvement of current P2P streaming protocols. As an important application, we propose a *Throughput Expectation Index (TEI)*, to facilitate the selection of high-bandwidth serving peers.

For each pair of ISPs, as there exists a daily evolutionary pattern for each of the regression coefficients in its throughput model, we summarize a daily intercept function (y-intercept) and a daily slope function, $\beta_0(t)$ and $\beta_1(t)$, respectively, where t represents different times in a day, by taking the average of coefficient values at the same time on different days. For example, Fig. 8.15 depicts the daily intercept and slope functions for intra-Telecom flows, summarized by averaging coefficients at the same hour during the week of December 17 — 23, 2006. We then define the following throughput expectation index:

$$TEI = \beta_0(t) + \beta_1(t) \times PEC(t). \tag{8.2}$$

TEI approximates the achievable inter-peer bandwidth between two peers across two ISPs (including two identical ISPs) at a specified time of a day. The computation of TEI not only captures all the deciding factors of inter-peer bandwidth — upload/download capacities at the upstream/downstream peer, concurrent upload/download load at the

upstream/downstream peer, and the ISPs both peers belong to — but also considers the temporal evolution of bandwidth availability at different times of a day. Therefore, it can be effectively utilized in peer selection at each peer, by *ranking* the candidate serving peers based on the computed TEI towards each of them. In more details, the TEI-assisted peer selection proceeds as follows:

The P2P streaming service provider derives the intercept and slope functions for each pair of ISPs, using the collected peer reports over a certain number of days (*e.g.*, one week). Upon bootstrapping a new peer, the intercept and slope functions of relevant ISP pairs, from each of the other ISPs to the ISP the peer belongs to, are loaded onto the peer. During the peer selection process, the peer obtains the following information from each of its candidate serving peers: IP address, upload capacity and the number of current upload flows. Then the peer calculates the per-flow end capacity of the potential P2P link from the candidate to itself, decides the intercept and slope functions to use (from its pre-loaded functions) by mapping the IP addresses of the candidate and itself to ISPs, and computes the TEI towards this candidate with y-intercept and slope values at the current moment. The peer ranks all candidate peers based on their derived TEIs. Then when the peer is deciding which media block to download from which candidate peer based on the exchanged buffer maps, it maximally retrieves available media blocks from the peers with the highest ranks.

Similar usage of TEI can be applied at a tracking server to select the best serving peers for a requesting peer. Note that such peer selections are performed without any intrusive measurements. Only a small number of intercept and slope functions need to be pre-loaded onto the peer, and a limited amount of information needs to be acquired from neighboring peers.

We further emphasize that in TEI-assisted peer selection, it is the relative *ranks* of peers computed by TEIs that are being used, instead of the absolute throughput values estimated with TEIs. This is because throughput levels may vary from day to day, but the daily throughput pattern persists for each ISP pair, and therefore the relative rank of peers may persist as well at a specified time on different days. This allows us to use the summarized intercept/slope functions and PEC values of end peers at a specified time to calculate the relative throughput ranks at the time.

To investigate the accuracy of the proposed TEI, we conduct a number of cross-validation experiments, by using intercept/slope functions summarized from the representative week (December 17 — 23, 2006) in TEI-assisted peer selection throughout the trace period. At each peer that appeared in the traces, we calculate the TEI towards each of its sending partners, and then compare their ranks computed by the TEIs with their *true ranks* based on the actual TCP throughput over the links. The experiments are divided into two parts.

First, we investigate the true rank of the best sending peer selected with TEI at each peer. Focusing on one snapshot at 9pm, December 18, 2006, Fig. 8.16(A) shows the distribution of this true rank at all the existing peers. At 70% of the peers, the TEI best peer coincides with the actual best sending peer with the largest throughput; at the majority of all peers, the TEI best peer ranks among top 3. Fig. 8.16(B) plots the evolution of the percentages of peers, at which the TEI best peer has a true rank no larger than 2 or 3, over the 10-week period of time. We observe that the former case achieves a peer percentage higher than 80% at all times, and the latter is consistently around 93%. During the throughput level shift around the earthquake scenario and the flash crowd scenario near Chinese New Year, the percentages represent larger fluctuations, but are

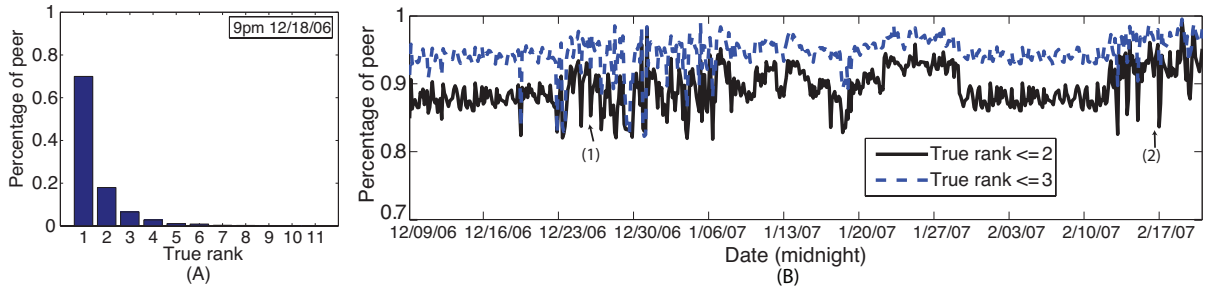


Figure 8.16: True rank distribution of the best sending peer selected with TEI. (1) Taiwan earthquake, (2) Chinese New Year Eve.

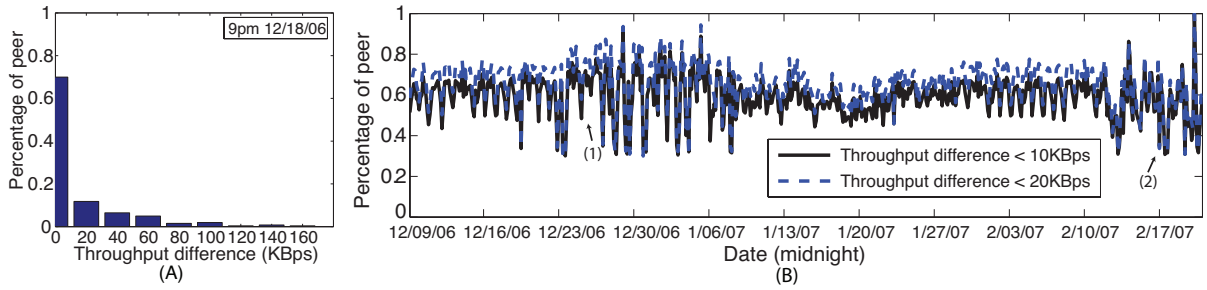


Figure 8.17: Distribution of throughput difference between flows from 5 top peers selected with TEI and flows from true top 5 peers. (1) Taiwan earthquake, (2) Chinese New Year Eve.

nevertheless quite satisfactory as well.

Next, we compare the sum of throughputs on P2P links from two peer groups at each receiving peer: (1) the 5 top sending peers selected with TEI, and (2) the true top 5 peers with largest throughputs. Fig. 8.17(A) shows the distribution of throughput difference between the two groups, at peers that existed at 9pm, December 18, 2006. The TEI selection achieves less than 10 KBps throughput difference at more than 70% peers. Furthermore, the difference is shown to be larger around the time of the two special scenarios in Fig. 8.17(B), and nevertheless, it is minor at most peers at most regular times, *i.e.*, 75% peers are subjected to a difference less than 20 KBps.

The above results exhibit that, while we are using intercept/slope functions summarized from only one week, the peer ranking mechanism of TEI works quite well throughout the trace period. This reflects the practical usefulness of TEI in capturing the persistent *relative* ranks of inter-peer bandwidths at each specific time on different days, without the need of intensive training using a large amount of historical data. A more elaborate usage of TEI may involve the retraining of the intercept/slope functions over time at the P2P streaming service provider, based on feedbacks from the peers about the accuracy of the current functions in evaluating high-bandwidth peers. As our goal is to show one effective application of our derived P2P streaming flow characteristics, we choose not to go into details of such practical protocol design.

8.5 Summary

Our study in this chapter represents the first attempt in the literature to characterize inter-peer bandwidth availability in modern large-scale P2P streaming networks. With abundant UUSee traces and using statistical means, we explore the critical factors that determine such achievable bandwidth, from both the end-host and ISP/area perspectives. In addition, we also explore the evolution of inter-peer bandwidth over time.

Our original discoveries in this study include: (1) The ISPs that peers belong to are more correlated to inter-peer bandwidth than their geographic locations; (2) Inter-ISP peering does not always constitute bandwidth bottlenecks, which is ISP specific; (3) There exist excellent linear correlations between peer last-mile bandwidth availability and inter-peer bandwidth within the same ISP, and between a subset of ISPs as well; (4) The evolution of inter-peer bandwidth between two ISPs exhibits a daily variation pattern, although the level of throughput values shifts from time to time; (5) During a

flash crowd scenario, the inter-peer bandwidth characteristics do not represent significant differences from those at regular times.

We have made use of the above insights in designing a throughput expectation index, which facilitates a new way of selecting high-bandwidth peers without any active and intrusive probing.

Chapter 9

Refocusing on Servers

As the essence of P2P streaming is the use of peer upload bandwidth to alleviate the load on dedicated streaming servers, so far our study, as well as most existing research, has focused on peer strategies: How should a mesh topology be constructed? What strategies can be designed to effectively utilize peer bandwidth? How do we select high-bandwidth peers? Nevertheless, in a practical P2P live streaming application, it is also important to provision adequate levels of stable upload capacities at the dedicated streaming servers, to guarantee the streaming quality in the streaming channels in case of peer instability and time-varying peer upload bandwidth availability.

In this chapter, we shift our focus to the streaming servers. Such refocusing on servers is motivated by our detailed analysis of 7 months (September 2006 - March 2007) and 400 GB worth of real-world traces from hundreds of streaming channels in UUSee. As all other state-of-the-art live streaming systems, in order to maintain a satisfactory and sustained streaming quality, UUSee has so far resorted to the practice of over-provisioning server capacities to satisfy the streaming demand from peers in each of its channels. Contrary to common belief, we have observed that available capacities on streaming servers are

not able to keep up with the increasing demand from hundreds of channels. Motivated by this observation, we advocate to explicitly allocate limited server capacities to each of the channels, in order to maximally utilize dedicated servers.

While it is certainly a challenge to determine how much bandwidth to provision on streaming servers to accommodate the streaming demand of all concurrent channels, the challenge is more daunting when we further consider the conflict of interest between P2P solution providers and ISPs. P2P applications have significantly increased the volume of inter-ISP traffic, which in some cases leads to ISP filtering. From our measurements, we have also observed a significant increasing trend of the volume of inter-ISP traffic in UUSee over time. Therefore, we seek to design an effective provisioning algorithm on servers with the awareness of ISP boundaries to minimize inter-ISP traffic.

Our proposal is *Ration*, an online server capacity provisioning algorithm to be carried out on a per-ISP basis. *Ration* dynamically computes the minimal amount of server capacity to be provisioned to each channel inside the ISP, in order to guarantee a desired level of streaming quality for each channel. With the analysis of our real-world traces, we have observed that the number of peers and their contributed bandwidth in each channel are dynamically varying over time, and significantly affect the required bandwidth from servers. *Ration* is designed to actively *predict* the bandwidth demand in each channel in an ISP with time series forecasting and dynamic regression techniques, utilizing the number of active peers, the streaming quality, and the server bandwidth usage within a limited window of recent history. It then proactively allocates server bandwidth to each channel, respecting the predicted demand and priority of channels.

To show the effectiveness of *Ration*, it has been implemented in streaming servers serving a mesh-based P2P streaming system. In a cluster of dual-CPU servers, the

system emulates real-world P2P streaming by replaying the scenarios captured by traces.

9.1 Evidence from Real-world Traces

Why shall we refocus our attention to dedicated streaming servers in P2P live streaming systems? Based on our study of 7-month worth of runtime traces from UUSee (400 GB of traces with more than 300 million unique IP addresses), we have made the following observations.

9.1.1 Insufficient "supply" of server bandwidth

First, we have observed that the server bandwidth becomes insufficient in the streaming system, as more channels are added over time. Such insufficiency has gradually affected the streaming quality, in both popular and less popular channels.

In order to show bandwidth usage over 7 months and at different times of a day within one figure, we choose to show all our 5-minute measurements on representative dates in each month. One such date, February 17 2007, is intentionally chosen to coincide with the Chinese New Year event, with typical flash crowds due to the broadcast of a celebration show on a number of the channels. Fig. 9.1(A) shows the total server bandwidth usage on the 150 streaming servers in UUSee network. We may observe that an increasing amount of server bandwidth has been consumed over time, but stabilizing in January 2007. This rising trend can be explained by the rapidly increasing number of channels deployed during this period, as shown in Fig. 9.1(B). The interesting phenomenon that such bandwidth usage has stabilized, even during the Chinese New Year flash crowd, has led to the conjecture that the total uplink capacity of all servers has been reached.

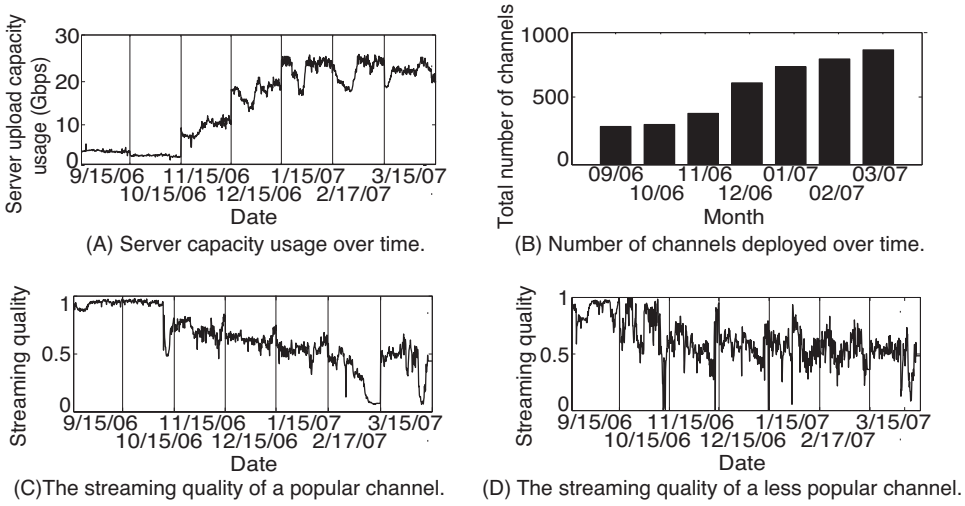


Figure 9.1: The evolution of server bandwidth, channels, and streaming quality over a period of 7 months.

The daily variation of server bandwidth usage coincides with the daily pattern of peer population, which peaks in the evenings.

Our conjecture that server capacities have saturated is confirmed when we investigate the streaming quality in each channel. The streaming quality in a channel at each time is evaluated as the *percentage of high-quality peers in the channel*, where a high-quality peer has a buffer count of more than 80% of the total size of its playback buffer. Recall the buffer count at each peer is the number of available blocks in its current playback buffer, as defined in Sec. 6.1. Representative results with a popular channel (*CCTV1*, with more than 10,000 concurrent users) and a less popular channel (*CCTV12*, with fewer than 1000 concurrent users) are shown in Fig. 9.1(C) and (D), respectively. The streaming quality of both channels has been decreasing over time, as server capacities are saturated. During the Chinese New Year flash crowd, the streaming quality of CCTV1 degraded significantly, due to the lack of bandwidth to serve a flash crowd of users in the channel.

Would it be possible that the lack of peer bandwidth contribution has overwhelmed the servers? As discussed in Sec. 6.1, the protocol in UUSee uses a number of algorithms to maximize peer upload bandwidth utilization, which in our opinion represents one of the state-of-the-art peer strategies in P2P streaming. The following back-of-the-envelope calculation with data from the traces may be convincing: At one time on October 15, 2006, about $100,000$ peers in the entire network have each achieved a streaming rate around 400 Kbps, by consuming a bandwidth level of 2 Gbps from the servers. The upload bandwidth contributed by peers can be computed as $100,000 \times 400 - 2,000,000 = 38,000,000$ Kbps, which is 380 Kbps per peer on average. This represents quite an achievement, as most of the UUSee clientele are ADSL users in China, with a maximum of 500 Kbps upload capacity.

Indeed, server capacities have increasingly become a bottleneck in real-world P2P live streaming solutions.

9.1.2 Increasing volume of inter-ISP traffic

The current UUSee protocol is not aware of ISPs. We now investigate the volume of inter-ISP traffic during the 7-month period, computed as the throughput sum of all links across ISP boundaries at each time, by mapping IP addresses to the ISPs using the mapping database from UUSee. Fig. 9.2 reveals that both the inter-ISP peer-to-peer and server-to-peer traffic have been increasing, quadrupled over the 7-month period, due to the increased number of channels and peers.

In China, the two nation-wide ISPs, *Netcom* and *Telecom*, charge each other based on the difference of inter-ISP traffic volume in both directions, and regional ISPs are charged based on traffic to and from the nation-wide ISPs. Both charging mechanisms have made

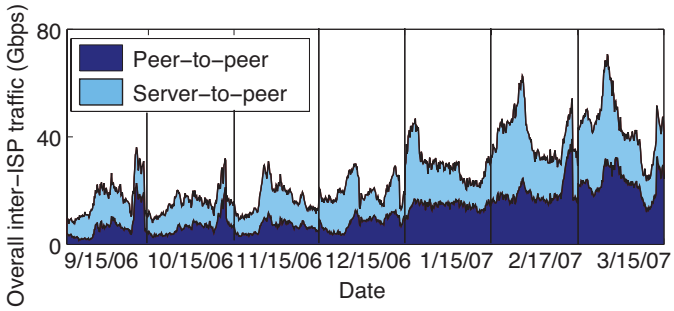


Figure 9.2: The volume of inter-ISP traffic increases over time.

it important for ISPs to limit inter-ISP traffic. Considering the large and persistent bandwidth consumption for live streaming, we believe that P2P streaming systems should be designed to minimize inter-ISP traffic, which remains one of our objectives in this chapter.

9.1.3 What is the required server bandwidth for each channel?

To determine the amount of server bandwidth needed for each channel, we wish to explore the relation among server upload bandwidth, the number of peers, and the achieved streaming quality in each channel. Fig. 9.3(A) illustrates the evolution of the three quantities for channel CCTV1 over a period of one week, from February 13 to 19, 2007. Though there appears to be a positive relation between server bandwidth and the streaming quality, and a negative relation between the peer population and streaming quality, these relationships are not strong and are far from consistent over time. For example, Fig. 9.3(B) plots the correlation of the quantities in a period of three days (February 13-15 2007). There does not exist any evident correlation in this period.

Nevertheless, if we focus on a shorter time scale, the correlation becomes more evident. For example, Fig. 9.3(C)-1 plots the correlation between server upload bandwidth usage and the streaming quality on February 13, which exhibits a positive square-root relation

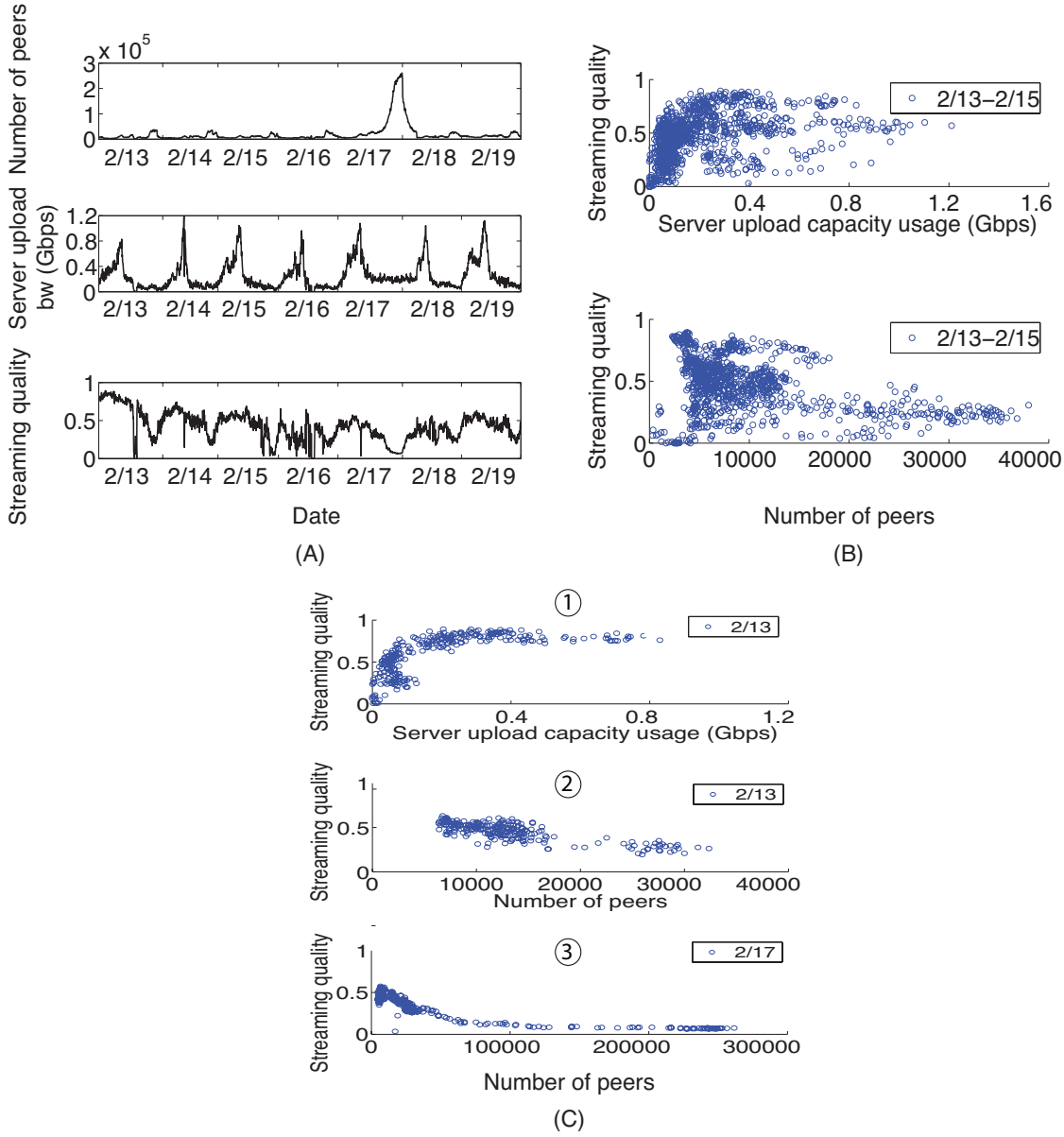


Figure 9.3: Relationship among server upload bandwidth, number of peers, and streaming quality for channel CCTV1.

between the two quantities. Meanwhile, a negative correlation is shown to exist between the number of peers and the streaming quality, in Fig. 9.3(C)-2. We have also observed that the shape of such short-term correlation varies from time to time. For example,

during the same time period on February 17, the relation between the number of peers and the streaming quality represents a reciprocal curve, as shown in Fig. 9.3(C)-3. We have observed from the traces that such variations exist in other channels as well, which can be attributed to the time-varying peer upload bandwidth availability in the channels.

All of our observations thus far point to the challenging nature of our problem at hand: how much server bandwidth should we allocate in each channel to assist peers in each ISP?

9.2 Ration: Online Server Capacity Provisioning

To answer the above questions, our proposal is *Ration*, an online server capacity provisioning algorithm to be carried out on a per-ISP basis, that dynamically assigns a minimal amount of server capacity to each channel to achieve a desired level of streaming quality.

9.2.1 Problem formulation

We consider a P2P live streaming system with multiple channels (such as UUSee). We assume that the tracking server in the system is aware of ISPs: when it supplies any requesting peer with information of new partners, it first assigns peers (or dedicated servers) with available upload bandwidth from the same ISP. Only when no such peers or servers exist, will the tracking server assign peers from other ISPs.

The focus of *Ration* is the dynamic provisioning of server capacity in each ISP, carried out by a designated server in the ISP. In the ISP that we consider, there are a total of M concurrent channels to be deployed, represented as a set $\mathcal{C}$. There are n^c peers in channel c, $\forall\ c \in \mathcal{C}$. Let s^c denote the server upload bandwidth to be assigned to channel c, and

q^c denote the streaming quality of channel c, *i.e.*, the percentage of high-quality peers in the channel that have a buffer count of more than 80% of the size of its playback buffer. Let U be the total amount of server capacity to be deployed in the ISP. We assume a priority level p^c for each channel c, that can be assigned different values by the P2P streaming solution provider to reflect the relative importance of the channels.

At each time t, *Ration proactively* computes the amount of server capacity s^c_{t+1} to be allocated to each channel c for time $t+1$, that achieves optimal utilization of the limited overall server capacity across all the channels, based on their priority and popularity (as defined by the number of peers in the channel) at time $t+1$. Such an objective can be formally represented by the optimization problem **Provision(t+1)** as follows ($\forall\ t = 1, 2, \ldots$), in which a *streaming quality function* F^c_{t+1} is included to represent the relationship among q^c, s^c and n^c at time $t+1$:

Provision(t+1):

$$\max \sum_{c \in \mathcal{C}} p^c n^c_{t+1} q^c_{t+1} \tag{9.1}$$

subject to

$$\begin{aligned} &\textstyle\sum_{c \in \mathcal{C}} s^c_{t+1} \leq U, \\ &q^c_{t+1} = F^c_{t+1}(s^c_{t+1}, n^c_{t+1}), \quad \forall\ c \in \mathcal{C}, \\ &0 \leq q^c_{t+1} \leq 1, s^c_{t+1} \geq 0, \quad \forall\ c \in \mathcal{C}. \end{aligned} \tag{9.2}$$

Weighting the streaming quality q^c_{t+1} of each channel c with its priority p^c, the objective function in (9.1) reflects our wish to differentiate channel qualities based on their priorities. With channel popularity n^c_{t+1} in the weights, we aim to provide better streaming qualities for channels with more peers. Noting that $n^c q^c$ represents the number of

high-quality peers in channel c, in this way, we guarantee that, overall, more peers in the network can achieve satisfying streaming qualities.

The challenges in solving **Provision(t+1)** at time t to derive the optimal values of $s_{t+1}^{c*},\ \forall\ c \in \mathcal{C}$, lie in (1) the uncertainty of the channel popularity n_{t+1}^{c}, *i.e.*, the number of peers in each channel in the future, and (2) the dynamic relationship F_{t+1}^{c} among q^c, s^c, and n^c of each channel c at time $t+1$. In what follows, we present our solutions to both challenges.

9.2.2 Active prediction of channel popularity

We first estimate the number of active peers in each channel c at the future time $t+1$, *i.e.*, $n_{t+1}^{c},\ \forall\ c \in \mathcal{C}$. Existing work has been modeling the evolution of the number of peers in P2P streaming systems based on Poisson arrivals and Pareto life time distributions (*e.g.*, [52]). We argue that these models represent ideal simplifications of real-world P2P live streaming systems, where peer dynamics are actually affected by many random factors. To dynamically and accurately predict the number of peers in a channel, we employ time series forecasting techniques. We treat the number of peers in each channel c, *i.e.*, $n_t^c, t = 1, 2, \ldots$, as an unknown random process evolving over time, and use the recent historical values to forecast the most likely values of the process in the future.

As the time series of channel popularity is generally non-stationary (*i.e.*, its values do not vary around a fixed mean), we utilize the *autoregressive integrated moving average* model, ARIMA(p,d,q), which is a standard linear predictor to tackle non-stationary time series. With ARIMA(p,d,q), a time series, $z_t, t = 1, 2, \ldots$, is differenced d times to derive a stationary series, $w_t, t = 1, 2, \ldots$, and each value of w_t can be expressed as the linear weighted sum of p previous values in the series, $w_{t-1}, \ldots, w_{t-p}$, and q previous random

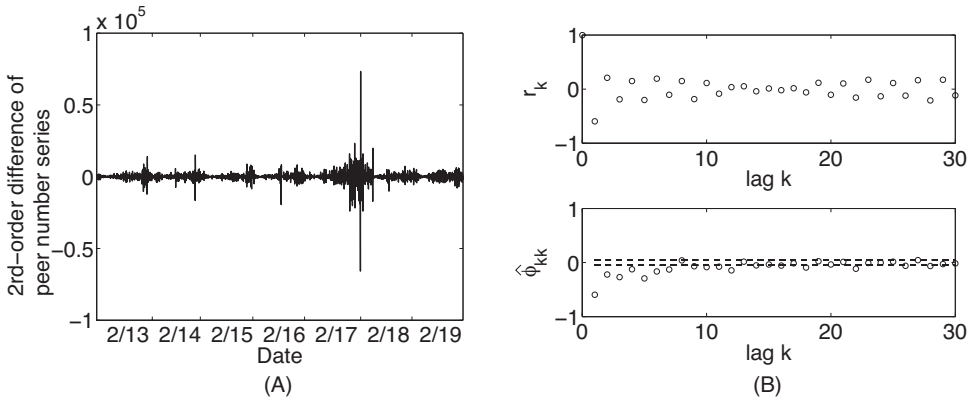


Figure 9.4: ARIMA model identification for channel popularity series of CCTV1, shown in Fig. 9.3(A).

errors, $a_{t-1}, \ldots, a_{t-q}$. The employment of an ARIMA(p,d,q) model involves two steps: (1) model identification, *i.e.*, the decision of model parameters p, d, q, and (2) model estimation, *i.e.*, the estimation of $p+q$ coefficients in the linear weighted summation.

In model identification, to determine the degree of differencing, d, a standard technique is to difference the time series as many times as is needed to produce stationary time series. We have therefore derived $d = 2$ for our time series $n_t^c, t = 1, 2, \ldots$, based on the observations that the second-order difference of the original time series for all the channels is largely stationary. For example, Fig. 9.4(A) shows the 2nd-order difference of the channel popularity time series of CCTV1, as given in Fig. 9.3(A), which is stationary with the mean of zero. To identify the values of p and q, the standard technique is to study the general appearance of the estimated autocorrelation and partial autocorrelation functions of the differenced time series ([22], pp. 187). For example, Fig. 9.4(B) plots the autocorrelation r_k and partial autocorrelation $\hat{\phi}_{kk}$ function values of the differenced channel popularity series in Fig. 9.4(A) up to lag $k = 30$. We observe that only r_1 is non-zero and $\hat{\phi}_{kk}$ tails off, which identifies $p = 0$ and $q = 1$ ([22], pp. 186). As we have generally made similar observations regarding channel popularity series for other

channels, we derive an ARIMA(0,2,1) model to use in our predictions.

Having identified an ARIMA(0,2,1) model, the channel popularity prediction at time $t+1$, $\bar{n}^c_{t+1}$ can be expressed as follows:

$$\bar{n}^c_{t+1} = 2n^c_t - n^c_{t-1} + a_{t+1} - \theta a_t, \tag{9.3}$$

where θ is the coefficient for the random error term a_t and can be estimated with a least squares algorithm. When we use (9.3) for prediction in practice, the random error at future time $t+1$, *i.e.*, a_{t+1}, can be treated as zero, and the random error at time t can be approximated by $a_t = n^c_t - \bar{n}^c_t$ [22]. Therefore, the prediction function is simplified to

$$\bar{n}^c_{t+1} = 2n^c_t - n^c_{t-1} - \theta(n^c_t - \bar{n}^c_t). \tag{9.4}$$

To dynamically refine the model for accurate prediction of popularity of a channel c over time, we propose to carry out the forecasting in a dynamic fashion: To start, the ARIMA(0,2,1) model is trained with channel popularity statistics in channel c in the most recent N_1 time steps, and the value of coefficient θ is derived. Then at each following time t, $\bar{n}^c_{t+1}$ is predicted using (9.4), and the confidence interval of the predicted value (at a certain confidence level, *e.g.*, 95%) is computed. When time $t+1$ comes, the actual number of peers, n^c_{t+1}, is collected and tested against the confidence bounds. If the real value lies out of the confidence interval and such prediction errors have occurred T_1 out of T_2 consecutive times, the forecasting model is retrained, and the above process repeats.

9.2.3 Dynamic learning of the streaming quality function

Next, we dynamically derive the relationship among streaming quality, server bandwidth usage, and the number of peers in each channel c, denoted as the streaming quality function F^c in (9.2), with a statistical regression approach.

From the traces, we have observed $q^c \propto (s^c)^{\alpha^c}$ at short time scales, where α^c is the exponent of s^c, *e.g.*, $q^c \propto (s^c)^{0.5}$ in Fig. 9.3(C)-1. We also observed $q^c \propto (n^c)^{\beta^c}$, where β^c is the exponent of n^c, *e.g.*, $q^c \propto (n^c)^{-1}$ in Fig. 9.3(C)-3. As we have made similar relationship observations from a broad trace analysis of channels over different times, we model the streaming quality function as

$$q^c = \gamma^c (s^c)^{\alpha^c} (n^c)^{\beta^c}, \tag{9.5}$$

where $\gamma^c > 0$ is a weight parameter. Such a function model is advantageous in that it can be transformed into a multiple linear regression problem, by taking logarithm at both sides:

$$\log(q^c) = \log(\gamma^c) + \alpha^c \log(s^c) + \beta^c \log(n^c).$$

Let $Q^c = \log(q^c)$, $S^c = \log(s^c)$, $N^c = \log(n^c)$, $\Gamma^c = \log(\gamma^c)$. We derive the following multiple linear regression problem

$$Q^c = \Gamma^c + \alpha^c S^c + \beta^c N^c + \epsilon^c, \tag{9.6}$$

where S^c and N^c are regressors, Q^c is the response variable, and ϵ^c is the error term. Γ^c, α^c, and β^c are regression parameters, which can be estimated with least squares algorithms.

As we have observed in trace analysis that the relationship in (9.5) is evident on short time scales but varies over a longer term, we dynamically re-learn the regression model in (9.6) for each channel c in the following fashion: To start, the designated server trains the regression model with collected channel popularity statistics, server bandwidth usage and channel streaming quality during the most recent N_2 time steps, and derives the values of regression parameters. At each following time t, it uses the model to estimate the streaming quality based on the used server bandwidth and the collected number of peers in the channel at t, and examines the fitness of the current regression model by comparing the estimated value with the collected actual streaming quality. If the actual value exceeds the confidence interval of the predicted value for T_1 out of T_2 consecutive times, the regression model is retrained with the most recent historical data.

We note that the signs of exponents α^c and β^c in (9.5) reflect positive or negative correlations between the streaming quality and its two deciding variables, respectively. Intuitively, we should always have $0 < \alpha^c < 1$, as the streaming quality could not be worse when more server capacity is provisioned, and its improvement slows down with more and more server capacity provided, until it finally reaches the upper bound of 1. On the other hand, the sign of β^c may be uncertain, depending on the peer upload bandwidth availability at different times: if more peers with high upload capacities (*e.g.*, Ethernet peers) are present, the streaming quality can be improved with more peers in the channel ($\beta^c > 0$); otherwise, more peer joining the channel could lead to a downgrade of the streaming quality ($\beta^c < 0$).

9.2.4 Optimal allocation of server capacity

Based on the predicted channel popularity and the most recently derived streaming quality function for each channel, we are now ready to proactively assign the optimal amount of server capacity to each channel for time $t+1$, by solving problem **Provision(t+1)** in (9.1). Replacing q^c with its function model in (9.5), we transform the problem in (9.1) into:

Provision(t+1)':

$$\max G \tag{9.7}$$

subject to

$$\sum_{c\in\mathcal{C}} s^c_{t+1} \leq U, \tag{9.8}$$

$$s^c_{t+1} \leq B^c_{t+1}, \qquad \forall\ c \in \mathcal{C}, \tag{9.9}$$

$$s^c_{t+1} \geq 0, \qquad \forall\ c \in \mathcal{C}, \tag{9.10}$$

where the objective function
$G = \sum_{c\in\mathcal{C}} p^c n^c_{t+1} q^c_{t+1} = \sum_{c\in\mathcal{C}} p^c \gamma^c (n^c_{t+1})^{(1+\beta^c)} (s^c_{t+1})^{\alpha^c}$, and $B^c_{t+1} = (\gamma^c (n^c_{t+1})^{\beta^c})^{-\frac{1}{\alpha^c}}$, denoting the maximal server capacity requirement for channel c at time $t+1$, that achieves $q^c_{t+1} = 1$.

The optimal server bandwidth provisioning for each channel, s^{c*}_{t+1}, $\forall\ c \in \mathcal{C}$, can be obtained with a water-filling approach. The implication of the approach is to maximally allocate the server capacity, at the total amount of U, to the channels with the current largest marginal utility, as computed with $\frac{dG}{ds^c_{t+1}}$, as long as the upper bound of s^c_{t+1} indicated in (9.9) has not been reached.

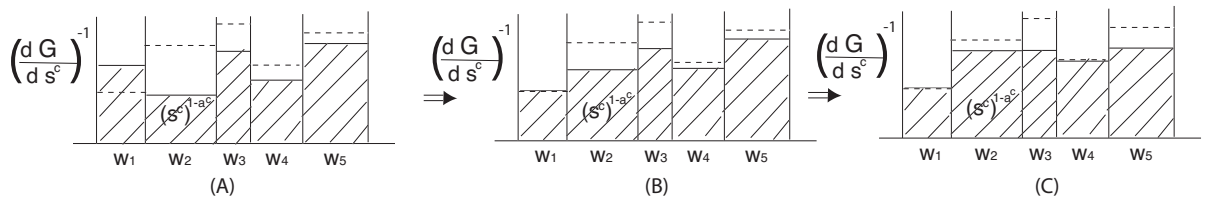


Figure 9.5: An illustration of the incremental water-filling approach with 5 channels.

In *Ration*, the server capacity assignment is periodically carried out to adapt to the changing demand in each of the channels over time. To minimize the computation overhead, we propose an *incremental water-filling approach*, that adjusts server capacity shares among the channels from their previous values, instead of a complete re-computation from the very beginning.

The incremental water-filling approach is given in Table 9.1. To better illustrate the idea of *water filling*, we utilize the reciprocal of marginal utility $\frac{dG}{ds^c_{t+1}}$, *i.e.*, $(\frac{dG}{ds^c_{t+1}})^{-1}$, in our algorithm description, and maximally assign server capacity to the channels with the current smallest value of $(\frac{dG}{ds^c_{t+1}})^{-1}$, equivalently.

We further explain the incremental water-filling approach with a 5-channel example in Fig. 9.5. In this figure, each channel is represented by one bin. The volume of water in bin c is $(s^c)^{(1-\alpha^c)}$, the width of bin c is $w_c = p^c\gamma^c\alpha^c(n^c)^{(1+\beta^c)}$, and thus the water level of the bin represents $(\frac{dG}{ds^c})^{-1} = \frac{(s^c)^{(1-\alpha^c)}}{w_c}$ for channel c. As $0 < \alpha^c < 1$, each bin has a maximal height, $\frac{(B^c_{t+1})^{(1-\alpha^c)}}{w_c}$, which is represented by the dashed line in each bin. The incremental water-filling approach starts with the optimal server capacity allocation at the current time t, *i.e.*, $s^c = s^{c*}_t$, $\forall\, c \in \mathcal{C}$ (Step 1 in Table 9.1). It first computes whether there exists any surplus of the overall provisioned server capacity, that occurs when not all the server capacity has been used with respect to the current allocation, *i.e.*, $U - \sum_{c\in\mathcal{C}} s^c > 0$, or the allocated capacity of some channel c exceeds its maximal

Table 9.1: Incremental water-filling approach

1. Initialize
   $s^c \leftarrow s_t^{c*},\ \forall\ c \in \mathcal{C}$.
2. Compute current surplus of server capacity
   $surplus = U - \sum_{c \in \mathcal{C}} s^c$.
   for all $c \in \mathcal{C}$
     if $s^c > B_{t+1}^c$
       $surplus = surplus + (s^c - B_{t+1}^c)$
     end if
   end for.
3. Allocate surplus to channels
   Compute $(\frac{dG}{ds^c})^{-1} = \frac{(s^c)^{1-\alpha^c}}{p^c \gamma^c \alpha^c (n^c)^{(1+\beta^c)}},\ \forall\ c \in \mathcal{C}$.
   while $surplus > 0$ and not all s^c has reached $B_{t+1}^c,\ \forall\ c \in \mathcal{C}$
     find the channel c_0 with the smallest value of $(\frac{dG}{ds^c})^{-1}$ and $s^{c_0} < B_{t+1}^{c_0}$.
     $s^{c_0} \leftarrow s^{c_0} + \delta$, $surplus \leftarrow surplus - \delta$.
     update $(\frac{dG}{ds^{c_0}})^{-1}$.
   end while
4. Adjust channel capacity assignment towards the same marginal utility
   if not all s^c has reached $B_{t+1}^c,\ \forall\ c \in \mathcal{C}$
     find channel c_{min} with the current smallest value of $(\frac{dG}{ds^c})^{-1}$ and $s^c < B_{t+1}^c$.
     find channel c_{max} with the current largest value of $(\frac{dG}{ds^c})^{-1}$.
     while $\| (\frac{dG}{ds^{c_{min}}})^{-1} - (\frac{dG}{ds^{c_{max}}})^{-1} \| > \epsilon$
       $s^{c_{min}} \leftarrow s^{c_{min}} + \delta$,$s^{c_{max}} \leftarrow s^{c_{max}} - \delta$.
       update $(\frac{dG}{ds^c})^{-1}$ for channels c_{min} and c_{max}.
       find channel c_{min} with the current smallest $(\frac{dG}{ds^c})^{-1}$ and $s^c < B_{t+1}^c$.
       find channel c_{max} with the current largest $(\frac{dG}{ds^c})^{-1}$.
     end while
   end if
5. $s_{t+1}^{c*} \leftarrow s^c,\ \forall\ c \in \mathcal{C}$.

server capacity requirement for time $t+1$, *i.e.*, $s^c > B^c_{t+1}$ (*e.g.*, the case that the water level goes above the maximal bin height for bin 1 in Fig. 9.5(A).) If so, the overall surplus is computed (Step 2 in Table 9.1) and allocated to the channels whose maximal server capacity requirement has not been reached, starting from the channel with the current lowest water level (Step 3 in Table 9.1). For the example in Fig. 9.5, the surplus part of the water in bin 1 in (A) is reallocated to bin 2 and bin 4, with the results shown in (B). After all the surplus has been allocated, the bandwidth assignment is further adjusted among the channels towards the same water level (marginal utility), in the fashion that the water (bandwidth) in the bin with the highest water level (largest value of $(\frac{dG}{ds^c})^{-1}$) is flooded into the bin that has the lowest water level (smallest value of $(\frac{dG}{ds^c})^{-1}$) and has not reached its maximal bin height yet (Step 4 in Table 9.1). For the example in Fig. 9.5(B), the water from bin 3 and 5 are filled into bin 2 and 4, until bin 4 reaches its maximal height and bins 2, 3, and 5 achieve the same water level, as shown in Fig. 9.5(C).

Such an incremental water-filling approach derives the optimal server capacity provisioning for all channels at time $t+1$, as established by the following theorem:

Theorem 1. *Given the channel popularity prediction* $n^c_{t+1},\ \forall\ c \in \mathcal{C}$*, and the most recent streaming quality function* $q^c_{t+1} = \gamma^c (s^c_{t+1})^{\alpha^c} (n^c_{t+1})^{\beta^c},\ \forall\ c \in \mathcal{C}$*, the incremental water-filling approach obtains an optimal server capacity provisioning across all the channels for time* $t+1$, i.e., $s^{c*}_{t+1},\ \forall\ c \in \mathcal{C}$*, which solves the problem* ***Provision(t+1)*** *in (9.1).*

Proof: Let $s^{c*}_{t+1},\ \forall\ c \in \mathcal{C}$ be an optimal solution to the optimization problem in (9.7). Introducing Lagrange multiplier λ for the constraint in (9.8), $\nu = (\nu^c,\ \forall\ c \in \mathcal{C})$ for the constraints in (9.9) and $\mu = (\mu^c,\ \forall\ c \in \mathcal{C})$ for the constraints in (9.10), we obtain the KKT conditions for the problem as follows (pp. 244, [23]):

$$\sum_{c\in\mathcal{C}} s_{t+1}^{c*} \le U,$$

$$\lambda^* \ge 0,$$

$$0 \le s_{t+1}^{c*} \le B_{t+1}^c, \nu^{c*} \ge 0, \mu^{c*} \ge 0, \ \forall\ c \in \mathcal{C},$$

$$\lambda^*(\sum_{c\in\mathcal{C}} s_{t+1}^{c*} - U) = 0, \tag{9.11}$$

$$\mu^{c*} s_{t+1}^{c*} = 0, \ \forall\ c \in \mathcal{C}, \tag{9.12}$$

$$\nu^{c*}(s_{t+1}^{c*} - B_{t+1}^c) = 0, , \ \forall\ c \in \mathcal{C} \tag{9.13}$$

$$-\frac{dG}{ds_{t+1}^c} + \lambda^* - \mu^{c*} + \nu^{c*} = 0, \ \forall\ c \in \mathcal{C}. \tag{9.14}$$

For $s_{t+1}^{c*} > 0$, we have $\mu^{c*} = 0$ from (9.12), and then $\frac{dG}{ds_{t+1}^c} = \lambda^* + \nu^{c*}$ from (9.14). Therefore, if further we have $s_{t+1}^{c*} < B_{t+1}^c$, we derive $\nu^{c*} = 0$ from (9.13) and then $\frac{dG}{ds_{t+1}^c} = \lambda^*$. As $\frac{dG}{ds_{t+1}^c} = \frac{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}{(s_{t+1}^c)^{1-\alpha^c}}$, $\gamma^c > 0$ and $0 < \alpha^c < 1$, we can derive, $\forall\ c \in \mathcal{C}$,

$$s_{t+1}^{c*} = \begin{cases} B_{t+1}^c & \text{if } \frac{1}{\lambda^*} \ge \frac{(B_{t+1}^c)^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}, \\ (\frac{p^c\gamma^c\alpha^c(n_{t+1}^c)^{1+\beta^c}}{\lambda^*})^{\frac{1}{1-\alpha^c}} & \text{if } \frac{1}{\lambda^*} < \frac{(B_{t+1}^c)^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}. \end{cases} \tag{9.15}$$

Notice that for all the channels with $0 < s_{t+1}^{c*} < B_{t+1}^c$, we have $\frac{1}{\lambda^*} = (\frac{dG}{ds_{t+1}^c})^{-1}$, which is the final water level for those bins whose maximal heights are not achieved, as illustrated in Fig. 9.5(C) (Note that $s_{t+1}^{c*} = 0$ only occurs at very special cases, such as $n_{t+1}^c \to 0$ or $\alpha^c \to 0$. In this case, the width of the bin corresponding to channel c is zero, and thus no water (bandwidth) will be assigned to the bin. We omit this special

case in our discussions.) We also notice that $\frac{(B_{t+1}^c)^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}$ is the maximal height for bin c. Therefore, the key to derive s_{t+1}^{c*} is to derive the optimal water level $\frac{1}{\lambda^*}$. If a bin's maximal height is below the optimal water level, the optimal server capacity share for the corresponding channel is its maximal server capacity requirement, *i.e.*, $s_{t+1}^{c*} = B_{t+1}^c$; otherwise, its allocated server capacity is what achieves $\frac{(s_{t+1}^{c*})^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}} = \frac{1}{\lambda^*}$.

To derive the optimal water level, from the starting water levels in the bins decided by the server capacity assignment at time t, we first make sure the overall server capacity at the amount of U is maximally filled into the bins, while no bin's maximal height is exceeded. Then, we decrease the high water levels by decreasing the server capacity assigned to the corresponding bins (as $\alpha^c < 1$, $(\frac{dG}{ds_{t+1}^c})^{-1}$ decreases with the decrease of s_{t+1}^c), and increase the low water levels by increasing the server capacity assigned to the corresponding bins, while guaranteeing the maximal height of each bin is never exceeded. When all bins reach the same water level, except those whose maximal heights have been reached, we have derived the optimal server capacity allocation for all channels for time $t+1$, as given in (9.15). □

9.2.5 Ration: the complete algorithm

Our complete algorithm is summarized in Table 9.2, which is periodically carried out on a designated server in each ISP. The only peer participation required is to have each peer in the ISP send periodical heartbeat messages to the server, each of which includes its current playback buffer count.

We note that in practice, the allocation interval is decided by the P2P streaming solution provider based on need, *e.g.*, every 30 minutes, and peer heartbeat intervals can be shorter, *e.g.*, every 5 minutes.

Table 9.2: Ration: the online server capacity provisioning algorithm

At time t, the designated server in each ISP

1. Peer statistics collection

Collect the number of active peers in each channel, $n_t^c,\ \forall\ c \in \mathcal{C}$, with peer heartbeat messages.

Collect per-peer buffer count statistics from the heartbeat messages, and derive the streaming quality for each channel, $q_t^c,\ \forall\ c \in \mathcal{C}$.

2. Channel popularity prediction for each channel $c \in \mathcal{C}$

Test if n_t^c is within the confidence interval of $\bar{n}_t^c$, the value predicted at time $t-1$.

If n_t^c lies out of the confidence interval for T_1 out of T_2 consecutive times, retrain the ARIMA(0,2,1) model with the most recent N_1 peer number statistics.

Predict the channel popularity at time $t+1$ by $\bar{n}_{t+1}^c = 2n_t^c - n_{t-1}^c - \theta(n_t^c - \bar{n}_t^c)$, where θ is the parameter in ARIMA(0,2,1).

$\rightarrow$ Channel popularity predictions for all channels are derived.

3. Learning the streaming quality function for each channel $c \in \mathcal{C}$

Estimate the streaming quality for time t with the current streaming quality function model: $\bar{q}_t^c = \gamma^c (s_t^c)^{\alpha^c} (n_t^c)^{\beta^c}$.

Test if the actual q_t^c is within the confidence interval of $\bar{q}_t^c$.

If q_t^c lies out of the confidence interval for T_1 out of T_2 consecutive times, retrain the regression model in Eq. (9.6) with statistics in the most recent N_2 times.

$\rightarrow$ The current streaming quality functions for all channels are derived.

4. Proactive server capacity provisioning for all the channels

Adjust server capacity assignment among all the channels with the incremental water-filling approach in Sec. 9.2.4.

$\rightarrow$ Optimal server capacity provisioning,$s_{t+1}^{c*},\ \forall\ c \in \mathcal{C}$,is derived.

We further briefly remark on the computational complexity of the algorithm in Table 9.2. The main computation for steps 2 and 3 lies at the training of ARIMA or the regression model, with least squares algorithms at $O(N^3)$ where N is the size of the training sample set. As both steps are carried out for each of the M channels, their complexity are at most $O(M{N_1}^3)$ and $O(M{N_2}^3)$, respectively. We generally need no more than $30 - 50$ samples to train an ARIMA(0,2,1) model, *i.e.*, $N_1 < 50$, and even less to derive the regression model (thus only a small amount of historical data needs to be maintained at the server for the execution of *Ration*). Further considering that the training is both only carried out when necessary, we conclude that the two steps incur low computational overhead in reality. At step 4, we have designed the incremental water-filling approach, which only involves local adjustments for channels that are affected. In summary, the algorithm involves low computational overhead, and can be carried out in a completely online fashion to adapt to the dynamics of P2P systems.

9.2.6 Practical implications

Finally, we discuss the practical application of *Ration* in real-world P2P live streaming systems.

Addressing various supply-demand relationships

In practical systems with unknown demand for server capacity in each ISP, *Ration* can make full utilization of the currently provisioned server capacity, U, and meanwhile provide excellent guidelines for the adjustment of U, based on different relationships between the supply and demand for server capacity.

If the P2P streaming system is operating at the over-provisioning mode in an ISP, *i.e.*,

the total deployed server capacity exceeds the overall demand from all channels to achieve the required streaming rate at their peers, *Ration* derives the minimal amount of server capacity needed for each channel c to achieve its best streaming quality, represented as $q^c = 1$. This is guaranteed by (9.9) in **Provision(t+1)'**, as the server capacity provisioned to each channel may not exceed the amount that achieves $q^c = 1$. When the P2P streaming solution provider discovers that the system is always operating at the over-provisioning mode, they may consider to reduce their total server capacity deployment in the ISP.

If the system is operating in a mode with tight supply-demand relations, *i.e.*, the total server capacity can barely meet the demand from all channels to achieve the best streaming qualities, *Ration* guarantees the limited server capacity is most efficiently utilized across the channels, respecting their demand and priority. With its water-filling approach, the preference in capacity assignment is based on marginal utility of each channel, $\frac{dG}{ds^c} = p^c n^c \frac{dq^c}{ds^c}$, as determined by the popularity and priority of the channel, and the marginal improvement of its streaming quality upon unit increase of server capacity. If the streaming solution provider wishes to improve the streaming quality of the channels in the ISP, they may further compute how much more server capacity to be added, using the derived streaming quality function in (9.5).

If the system is operating with extremely tight supply-demand relations, *e.g.*, the flash crowd scenario, most server capacity is assigned to the one or few channels that are involved in the flash crowd, and most of the other channels are starving with no or very little server bandwidth. Upon detecting this, our algorithm can trigger the deployment of backup server resources. Similarly, the extra amount to add can be computed with the current streaming quality function derived for the respective channels.

Making decisions on channel deployment in each ISP

In addition, with *Ration*, the P2P streaming solution provider can dynamically make decisions on channel deployment in each ISP, when it is not possible or necessary to deploy every one of the hundreds or thousands of channels in each ISP. When a channel is not allocated any server capacity due to very low popularity or priority during a period of time, the channel is not to be deployed in the ISP during this time.

Deploying server capacities

Finally, we note that the server capacity provisioning in each ISP can be implemented in practice with a number of servers deployed, with the number decided by the total capacity to be provisioned and the upload capacity of each server. Inside each ISP, the servers can be further deployed in different geographic areas, and the derived server capacity provisioning for each channel can be distributed among several servers as well, in order to achieve load balancing and streaming delay minimization.

9.3 Experimental Evaluations with Trace Replay

Our evaluation of *Ration* is based on its implementation in a multi-ISP mesh-based P2P streaming system, which replays real-world streaming scenarios captured by the traces.

The P2P streaming system is implemented in C++ on a high-performance cluster of 50 Dell 1425SC and Sun v20z dual-CPU servers, interconnected by Gigabit Ethernet. On this platform, we are able to emulate hundreds of concurrent peers on each cluster server, and emulate all network parameters, such as node/link capacities. Actual media streams are delivered over TCP connections among the peers, and control messages are sent by

UDP. The platform supports multiple event-driven asynchronous timeout mechanisms with different timeout periods, and peer joins and departures are emulated with events scheduled at their respective times.

The P2P streaming protocol we implemented includes both the standard pull protocol and the unique algorithms employed by UUSee, as introduced in Sec. 6.1. Without loss of generality, we deploy one server for each ISP, implementing both the tracking server and streaming server functions. *Ration* is also implemented on each of the servers, with 800 lines of C++ code. The server capacity allocation for each channel is implemented by limiting the total number of bytes sent over the outgoing connections from the server for the channel in each unit time.

Our experiments are carried out on realistic replays of the traces. We emulate peer dynamics based on the evolution of the number of peers in each channel from the traces: when the number of peers rises between two consecutive time intervals, we schedule a corresponding number of peer join events during the interval; when the number of peers decreases, peer departure events are scheduled for a corresponding number of randomly selected peers. Upon arrival, each peer acquires 30 initial upstream peers, and the P2P topology evolves based on the same peer selection protocol as UUSee's. The node upload capacities are emulated using values from the traces, which follow a heavy-tail distribution in the major range of 50 Kbps to 10 Mbps. The streaming rate of each channel is 400 Kbps, with the streams divided into 1-second blocks for distribution. The size of playback buffer on the peers is set to 30 seconds. Each peer reports its buffer count to the server in its ISP every 20 seconds, and the server processes them and adjusts capacity allocation every 60 seconds.

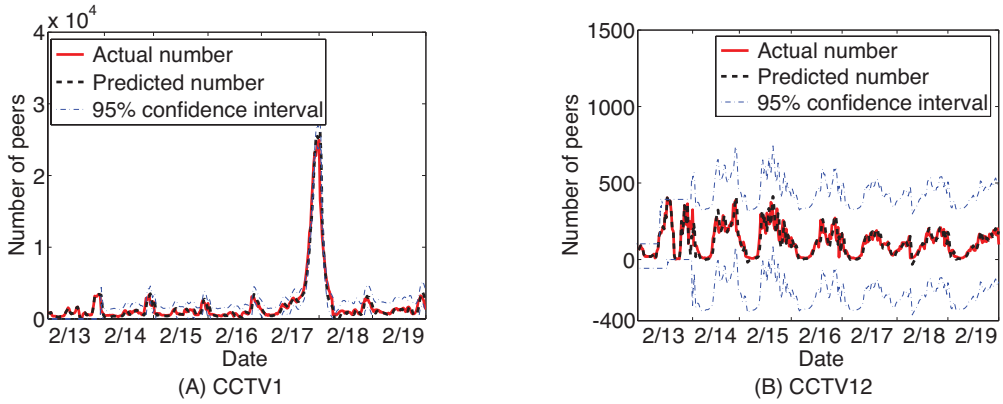


Figure 9.6: Prediction of the number of peers with ARIMA(0,2,1).

9.3.1 Performance of Ration components

We first examine the effectiveness of each composing algorithm in *Ration*. In this set of experiments, we focus on the streaming inside one ISP, with one server of 80 Mbps upload capacity and 5 channels. We use the peer number statistics of 5 channels from the traces, *CCTV1*, *CCTV4*, *CCTV2*, *CCTV7*, and *CCTV12*, in one ISP (China *Telecom*) during the week of February 13 – 19, 2007. To expedite our experiments, each peer number time series from the traces is sampled, such that the evolution of the P2P system in each day is emulated within half an hour. The 5 channels have a regular instantaneous number of peers at the scale of 2000, 500, 400, 150 and 100, respectively. The statistics of CCTV1 and CCTV4 also captured the flash crowd scenario on February 17, 2007, when the Chinese New Year celebration show was broadcast on the two channels.

Prediction of the number of peers

Fig. 9.6 presents the results of prediction with ARIMA(0,2,1) for the popular channel CCTV1 and the unpopular channel CCTV12, respectively. In the dynamic prediction, the training set size is $N_1 = 30$, and the error count parameters are $T_1 = 8$ and $T_2 = 10$.

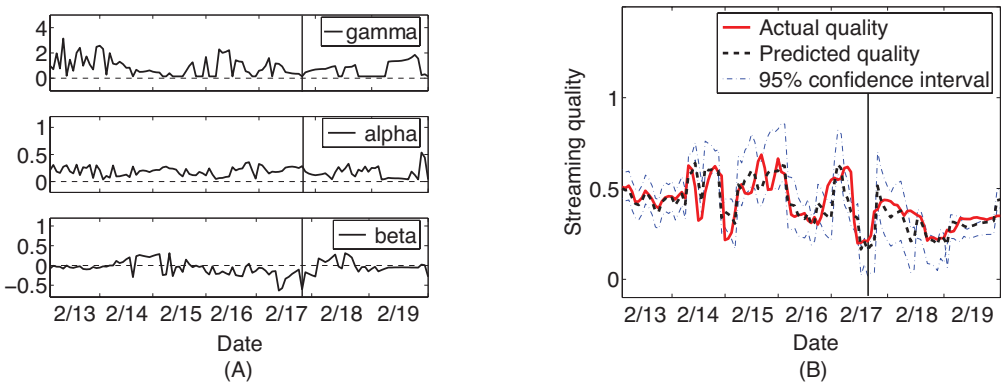


Figure 9.7: Dynamic learning of the streaming quality function for CCTV1.

The predicted numbers for both channels largely coincide with the actually collected number of peers, both at regular times and during the flash crowd, no matter whether the prediction confidence interval is large or small at different times. This validates the correctness of our model identification, as well as the accuracy of our dynamic prediction.

Dynamic streaming quality function

Fig. 9.7(A) plots the derived parameter values for the dynamic streaming quality function of CCTV1. In the dynamic regression, the training set size is $N_2 = 20$, the error count parameters are $T_1 = 8$ and $T_2 = 10$. We see that γ^c is all positive, the values of α^c are always within the range of $0 - 1$, and β^c may be positive or negative at different times. We have observed similar results with the derived streaming quality functions of other channels. This validates our analysis in the last paragraph of Sec. 9.2.3. During the flash crowd scenario, which hereinafter is marked with a vertical line in the figures, β^c is significantly below zero, revealing a negative impact on the streaming quality with a rapidly increasing number of peers in the channel.

Fig. 9.7(B) plots the actually measured streaming quality in the channel against its

estimated value, calculated with the derived streaming quality function at each time. The actual streaming quality closely follows the predicted trajectory at most times, including the flash crowd scenario, which exhibits the effectiveness of our dynamic regression.

Optimal provisioning among all channels

We now investigate the optimal server capacity provisioned to different channels over time. In this experiment, we focus on examining the effects of channel popularity on capacity allocation, and set the priorities for all 5 channels to the same value of 1.

Fig. 9.8(A) and (B) show the server capacity allocated to each of the 5 channels, and their actually achieved streaming quality at different times. We observe that, generally speaking, the higher the channel's popularity is, the more server capacity it is assigned. This can be explained by the marginal utility of the channels used in the water-filling allocation of *Ration*, $\frac{dG}{ds^c} = p^c n^c \frac{dq^c}{ds^c} = \frac{p^c \gamma^c \alpha^c (n^c)^{(1+\beta^c)}}{(s^c)^{1-\alpha^c}}$. As $\beta^c > -1$ is observed in our previous experiment, the marginal utility is positively correlated with the number of peers, and thus the more popular channel is assigned more server capacity.

On the other hand, in Fig. 9.8(B), we do not observe evident correlation between the channel popularity and its achieved streaming quality, as the latter is decided by both the allocated server capacity (positively) and the number of peers (positively or negatively at different times). Nevertheless, we show that our water-filling assignment achieves the best utilization of the limited overall server capacity at all times, with a comparison study to a proportional allocation approach.

The proportional allocation approach goes as follows: At each time t, instead of using water-filling, the server capacity is proportionally allocated to the channels, based only on their predicted number of peers for time $t+1$. Fig. 9.8(C) shows that the most popular

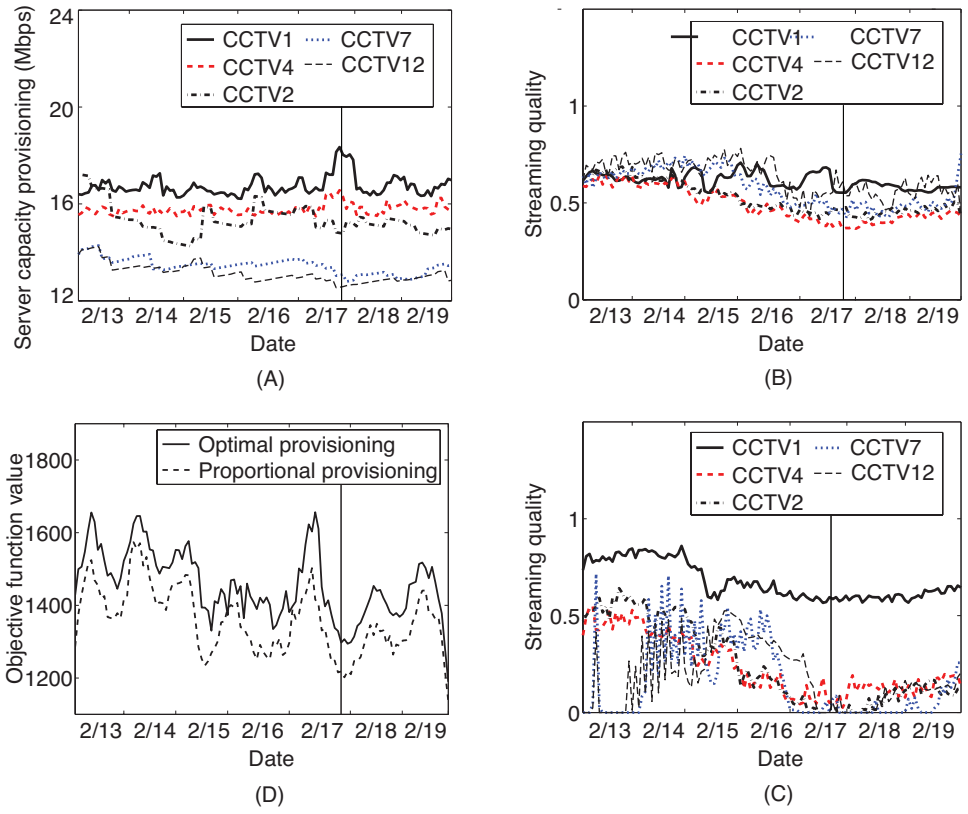


Figure 9.8: Server capacity provisioning for 5 non-prioritized channels: (A) Server capacity provisioning with *Ration*, (B) Streaming quality achieved with *Ration*, (C) Streaming quality achieved with proportional allocation, (D) Comparison of objective function values.

channel, CCTV1, achieves better streaming quality with this proportional allocation as compared to that in Fig. 9.8(B), at the price of downgraded quality for the other channels, especially during the flash crowd. This is because CCTV1 now obtains more than half of the total server capacity at regular times, and almost all during the flash crowd scenario.

With the streaming quality results in Fig. 9.8(B) and (C), we compute the values of the objective function of **Provision(t+1)** in (9.1), and plot them in Fig. 9.8(D). Given a same priority for all the channels, the value of the objective function at each time represents the total number of peers in all the channels that achieve satisfying streaming

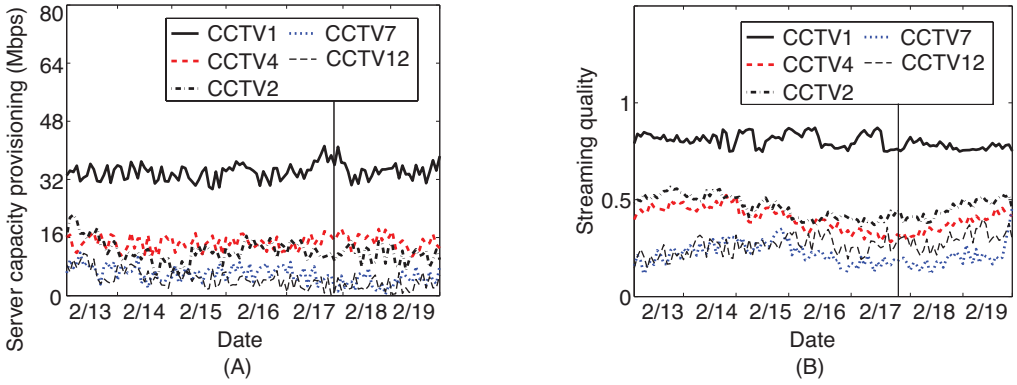


Figure 9.9: Server capacity provisioning for 5 prioritized channels with *Ration*.

rates at the time. The values from the proportional allocation are consistently lower than those achieved with our water-filling approach, exhibiting the optimality of the server capacity utilization with *Ration*.

Effectiveness of channel prioritization

In the next experiment, we investigate the effect of channel prioritization on server capacity provisioning with *Ration*, by setting 3 priority levels: $p^c = 500$ for CCTV1, $p^c = 200$ for CCTV4 and CCTV2, and $p^c = 50$ for CCTV7 and CCTV12.

Comparing its results in Fig. 9.9(A) to those in Fig. 9.8(A), we observe further differentiated server capacities among the channels, where the channel with the highest priority and popularity, CCTV1, is allocated much more capacity than the others. In Fig. 9.9(B), we also observe differentiated streaming qualities among channels based on their priority levels. These demonstrate the effectiveness of channel prioritization in *Ration*, which facilitates the streaming solution provider to differentiate services across channels, when the supply-demand relation of server capacity is tight in the system.

9.3.2 Effectiveness of ISP-aware server capacity provisioning

Next, we evaluate *Ration* in multi-ISP streaming scenarios. 4 ISPs are emulated by tagging servers and peers with their ISP IDs. Again, 5 channels, *CCTV1*, *CCTV4*, *CCTV2*, *CCTV7*, *CCTV12*, are deployed in the ISPs, with peer number statistics in each ISP extracted from those in 4 China ISPs, *Telecom*, *Netcom*, *Unicom* and *Tietong*, from the traces. While a fixed overall server capacity is used in the previous experiments, in the following experiments, we do not cap the server capacity, but derive with *Ration* the minimal amount of overall server capacity needed to achieve the best streaming qualities for all the channels in the system (*i.e.*, $q^c = 1, \forall\ c \in \mathcal{C}$), which is referred to as U_B hereinafter. At each time during the dynamic provisioning, U_B is derived by summing up the upper bound of server capacity required for each of the channels, as given in (9.9), at the time. Our focus is to compare the total server capacity U_B required when ISP awareness is in place and not, and the inter-ISP traffic that is caused. The channels are not prioritized in this set of experiments.

Without ISP awareness

In the first experiment, we deploy one server in the system, and stream with a peer selection protocol that is not ISP-aware. The overall server capacity U_B used on the server over time is shown in Fig. 9.10(A), and the total inter-ISP P2P traffic in the system is plotted in Fig. 9.10(B).

With full ISP awareness

In the second experiment, we deploy one server in each ISP, and constrain all streaming traffic inside each ISP by fully ISP-aware peer selection, *i.e.*, peers are only assigned

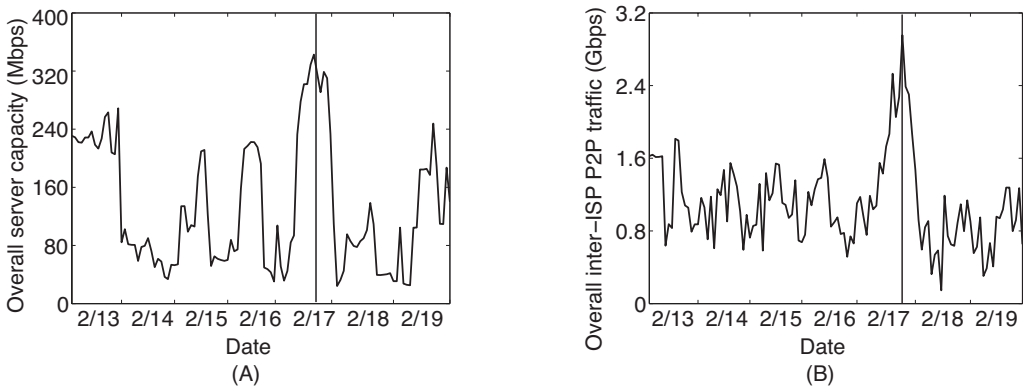


Figure 9.10: P2P live streaming for 5 channels in 4 ISPs: without ISP awareness.

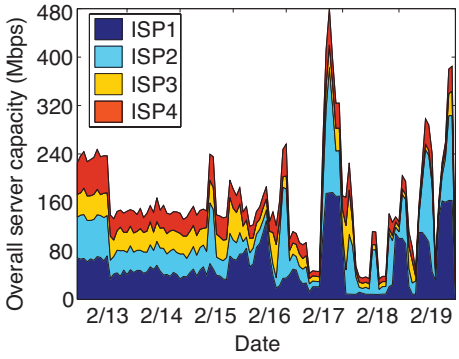


Figure 9.11: P2P live streaming for 5 channels in 4 ISPs: with full ISP awareness.

partners inside the ISP. The server capacity used on the server in each ISP is illustrated with the area plot in Fig. 9.11. Comparing Fig. 9.11 to Fig. 9.10(A), we can see that the overall server capacity usage in the system does not increase much when the traffic is completed restricted inside each ISP with per-ISP server capacity deployment. The difference is only larger during the flash crowd, the reason for which, we believe, is that it becomes hard for peers to identify enough supplying peers inside the ISP when the total number of peers soars, and thus they have to resort to the server.

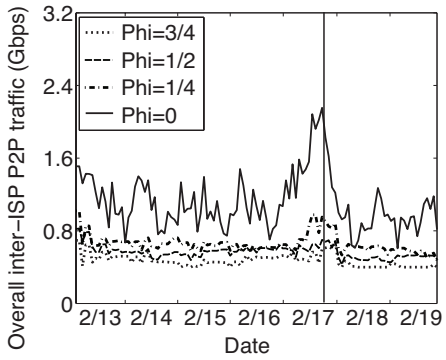


Figure 9.12: Server capacity provisioning vs. inter-ISP traffic: a tradeoff.

Tradeoff between server capacity and inter-ISP traffic

In the final experiment, we provision a total server capacity in the system that is between the amount used in Fig. 9.10(A) and that used in Fig. 9.11, and examine the resulting inter-ISP traffic. Specifically, let the overall server capacity usage shown in Fig. 9.10(A) be U_{Bmin} and that shown in Fig. 9.11 be U_{Bmax}. We reduce the server capacity provisioned on each server in each ISP, such that the overall server capacity at each time is at the value of $U_{Bmin} + \phi(U_{Bmax} - U_{Bmin})$ at the time. In this case, peers are allowed to connect to servers/peers across ISPs if they fail to acquire sufficient streaming bandwidth within the ISP.

The experiment is repeated by setting ϕ to $\frac{3}{4}$, $\frac{1}{2}$, $\frac{1}{4}$ or 0, that represent different levels of the total server capacity. The results in Fig. 9.12 show an increase of inter-ISP traffic with the decrease of server capacity provisioning. Further comparing the $\phi = 0$ case in Fig. 9.12 to Fig. 9.10(B), we observe that while the total server capacity is the same U_{Bmin} in both cases, a smaller amount of inter-ISP P2P traffic is involved with the ISP-aware peer selection than without any ISP awareness.

9.4 Summary

As the first in the literature, we present detailed measurements of server capacity utilization in a real-world P2P streaming system, and an online server capacity provisioning mechanism to address the dynamic demand in multi-ISP multi-channel P2P live streaming systems. In practice, we believe that it is important to refocus our attention on dedicated streaming servers: based on our detailed analysis of 7 months of traces from UUSee, available server capacities are not able to keep up with the increasing demand in such a practical system, leading to a downgrade of peer streaming quality. Emphasizing on practicality, our proposed algorithm, *Ration*, is able to dynamically predict the demand in each channel, using an array of dynamic learning techniques, and to proactively provision optimal server capacities across different channels. With full ISP awareness, *Ration* is carried out on a per-ISP basis, and is able to guide the deployment of server capacities and channels in each ISP to maximally constrain P2P traffic within ISP boundaries. Our performance evaluation of *Ration* features the realistic replay of real-world streaming traffic from our traces. We show that *Ration* lives up to our expectations to effectively provision server capacities according to the demand over time.

Chapter 10

Concluding Remarks

10.1 Conclusions

The main focus of this thesis is to improve key protocols in large-scale mesh-based P2P live streaming systems from both theoretical and practical perspectives.

From the more theoretical perspective, we focus on the optimization of the utilization of the crucial bandwidth resources in P2P streaming. In a single-overlay system, we model the optimal peer selection and bandwidth allocation problem into a linear optimization model, which minimizes end-to-end streaming latencies, and design a fully distributed algorithm to carry out the optimization. In the more practical scenario without any assumption on *a prior* knowledge of link or node bandwidth, we model the streaming rate satisfactory problem into a linear feasibility problem, and design an efficient distributed protocol to achieve it. In a realistic multi-overlay system, we resolve the bandwidth competitions among multiple coexisting streaming overlays by modeling them into a distributed collection of dynamic bandwidth auction games. Solidly established on the foundation of game theory, we design effective auction strategies, and rigidly

prove their convergence to a desirable optimal Nash equilibrium in realistic asynchronous environments.

From the more practical perspective, we extensively measure and analyze a commercial large-scale P2P live streaming system over a long period of time. Using a unique measurement methodology, in that we implement measurement facilities into the P2P client software, we collect a large volume of consecutive traces throughout the trace period. We are able to acquire a thorough and in-depth understanding of the large-scale mesh-based streaming, and derive many novel in-depth insights that is not possible with previous measurement methodologies. We have thoroughly investigated the topological properties of large-scale live streaming meshes, extensively characterized inter-peer bandwidths using statistical means, and carefully studied the importance of server capacity provisioning in P2P streaming. Based on a synergistic application of measurement insights and theoretical models, we have designed effective and practical algorithms to improve the key protocols in P2P streaming, *e.g.*, peer selection and deployment of streaming server capacities. We have also considered the rising tension between P2P applications and ISPs in our proposals, by effectively constraining the P2P traffic within ISP boundaries with server capacity deployment.

10.2 Future Directions

10.2.1 Possible enhancement of minimum-delay P2P streaming modeling

In addressing the minimum-delay peer selection and streaming rate allocation problem in Chapter 3, we have modeled a linear program which minimizes average end-to-end link

delays for the unicast flows from the server to each of the receivers (*i.e.*, model (3.1)). An intuitive thought about the optimization model is that, the end-to-end latency of a unicast flow should be formulated as the maximum of the end-to-end latencies of all its fraction flows, instead of the average, *i.e.*, the objective function can be formulated as $\min \max_{p \in P} \sum_{(i,j):(i,j) \text{ on } p} c_{ij}$. We have modeled the average end-to-end latency in our objective function due to the following considerations. (1) We are able to derive a nice linear formulation in (3.1), which is indeed a standard minimum-cost flow problem. Besides, when c_{ij}'s ($\forall\ (i,j) \in A$) are of similar magnitude, we can understand the objection function as the minimization of the average hop count from the streaming server to the receiver. (2) The minimization of the average end-to-end delay achieves similar results, while not exactly the same, as the minimization of maximum end-to-end latency: the rates are allocated in such a way that high latency links, such as satellite and transcontinental links, are avoided as much as possible whenever there are available bandwidths on low latency links. The solvability of the optimization problem minimizing the maximum delays and achievability of the solution with nice distributed structures require further studies.

10.2.2 Statistical traffic analysis of real-world networking systems

We believe a thorough understanding of existing solutions is an important prerequisite towards the design of the next generation Internet systems. Built on our experience of collaborating with UUSee Inc., we would like to continue with such large-scale measurements, by establishing close research collaborations with commercial developments. Our objective is to obtain in-depth insights from careful investigations of real-world systems

over long periods of time. We believe statistical techniques represent powerful tools towards in-depth analysis of Internet systems, including simple linear, multivariate linear, and nonlinear regressions, as well as time series structural analysis techniques. It will be intriguing research to apply these statistical techniques in the characterization of modern large-scale content distribution systems, such as novel P2P Internet applications, Content Distribution Networks, and mobile wireless systems.

10.2.3 Addressing the conflict between ISPs and P2P applications

With the widespread use of P2P applications over the Internet and the shift of bandwidth load on ordinary users, the conflict between P2P solution providers and Internet Service Providers has become a major challenge, that leads to the heated debate of *network neutrality* in the US Congress. In this thesis, we have made our first effort in addressing such a conflict in P2P streaming applications by constraining the P2P traffic within ISP boundaries with server capacity deployment. However, there still exist a lot of interesting research topics in this area to be explored: First, what may actually be the current scale of inter-ISP P2P traffic, including P2P file sharing, streaming and VoIP applications? Large-scale measurement studies are required towards an in-depth understanding to address this question. Second, how shall we characterize the economical conflict between ISPs and P2P solution providers, that might motivate them to collaborate to generate a solution? Game theory may come into play to provide strategic solutions for both parties. Third, if we wish to constrain P2P traffic within ISP boundaries by server deployment, what caching policy shall we deploy on such servers to guarantee service quality of different P2P applications? Statistical techniques and optimization theories may constitute

powerful tools for solving these problems.

10.2.4 Optimized games for network resource allocation

Game theory has represented a novel and intriguing way for networking algorithm design, that characterizes the challenging selfish nature of participants in network applications. The Nash equilibrium of a networking game represents the stable operating point of node behavior over time. However, in most cases, the Nash equilibrium is not an optimized working point for the system, *i.e.*, the existence of the *price of anarchy*. In this thesis, we have proposed an auction game mechanism that achieves an optimal Nash equilibrium. In generalization, we believe that an algorithm designed based on game theory is most valuable when an efficient Nash equilibrium, that represents global optimal properties, can be achieved. It is an intriguing challenge to design effective incentives or pricing mechanisms to guide the node behavior towards such an optimal operating point, or a best approximation to the optimum. Together with the decentralized nature of game play, the networking game will then essentially constitute a distributed optimization algorithm, or approximation algorithm, to solve a global performance optimization problem. It constitutes an intriguing research topic to incorporate the insights offered by both optimization theory and game theory, to design efficient, practical, and decentralized algorithms.

10.2.5 Stochastic performance modeling of P2P applications

Significant research efforts have been devoted towards the stochastic modeling and performance analysis of BitTorrent-like protocols for file sharing, while little analytical attention has been paid to P2P live streaming applications. While practical P2P streaming

applications have largely employed simple protocol designs, it would be interesting to stochastically model the behavior of such P2P systems, in order to improve existing design choices. Rather than making simplistic assumptions that are far from realistic (such as Poisson peer arrival in live streaming systems), it will be more convincing to derive dynamics models—such as peer arrival and lifetime distributions—from in-depth measurement investigations of real-world P2P systems. Combining large-scale measurements with stochastic tools, such modeling efforts will lead to instrumental theoretical insights that may essentially affect the design of practical systems.

Bibliography

[1] All-Sites-Pings for PlanetLab. http://ping.ececs.uc.edu/ping/.

[2] Joost. http://www.joost.com/.

[3] Log-normal distribution. http://mathworld.wolfram.com/LogNormalDistribution.html.

[4] PlanetLab IPerf. http://jabber.services.planet-lab.org/php/iperf/.

[5] PPLive. http://www.pplive.com/en/index.html.

[6] PPStream. http://www.ppstream.com/.

[7] QQLive. http://tv.qq.com/.

[8] SopCast. http://www.sopcast.org/.

[9] TVAnts. http://www.tvants.com/.

[10] UUSee. http://www.uusee.com/.

[11] VVSky. http://www.vvsky.com/.

[12] Zattoo. http://zattoo.com/.

[13] M. Adler, R. Kumar, K. W. Ross, D. Rubenstein, T. Suel, and D. D. Yao. Optimal Peer Selection for P2P Downloading and Streaming. In *Proc. of IEEE INFOCOM*, March 2005.

[14] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[15] A. Ali, A. Mathur, and H. Zhang. Measurement of Commercial Peer-To-Peer Live Video Streaming. In *Proc. of Workshop in Recent Advances in Peer-to-Peer Streaming*, August 2006.

[16] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz. Analyzing Stability in Wide-Area Network Performance. In *Proc. of ACM SIGMETRICS*, 1997.

[17] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. of ACM SIGCOMM*, August 2002.

[18] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.

[19] D. P. Bertsekas and D. A. Castanon. The Auction Algorithm for the Transportation Problem. *Annals of Operations Research*, 20:67–96, 1989.

[20] D. P. Bertsekas and R. Gallager. *Data Networks, 2nd Ed.* Prentice Hall, 1992.

[21] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[22] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control (Third Edition)*. Prentice Hall, 1994.

[23] S. Boyd. *Convex Optimization*. Cambridge University Press, 2004.

[24] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[25] R. Christensen. *Analysis of Variance, Design, and Regression: Applied Statistical Methods*. Chapman and Hall/CRC, 1996.

[26] Y.-H. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *Proc. of ACM SIGMETRICS*, June 2000.

[27] B.-G. Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz. Characterizing Selfishly Constructed Overlay Routing Networks. In *Proc. of IEEE INFOCOM*, March 2004.

[28] Y. Cui, Y. Xue, and K. Nahrstedt. Optimal Resource Allocation in Overlay Multicast. In *Proc. of 11th International Conference on Network Protocols (ICNP 2003)*, Atlanta, Georgia, USA, November 2003.

[29] Y. Cui, Y. Xue, and K. Nahrstedt. Max-min Overlay Multicast: Rate Allocation and Tree Construction. In *Proc. of the 12th IEEE International Workshop on Quality of Service (IWQoS 2004)*, June 2004.

[30] G. B. Dantzig. *Linear Programming and Extensions*. Prentice University Press, Princeton, New Jersey, 1963.

[31] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming Live Media over a Peer-to-Peer Network. Technical report, Standford Database Group 2001-20, August 2001.

[32] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a Network Creation Game. In *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, July 2003.

[33] A. V. Fiacco and G. P. McCormick. Nonlinear Programming: Sequential Unconstrained Minimization Techniques. *Reprinted in SIAM Classics in Applied Mathematics Series*, 4, 1990.

[34] N. Garg, R. Khandekar, K. Kunal, and V. Pandit. Bandwidth Maximization in Multicasting. In *Proc. of the 11th Annual European Symposium on Algorithms*, Budapest, Hungary, September 2003.

[35] D. Garlaschelli and M. I. Loffredo. Patterns of Link Reciprocity in Directed Networks. *Physical Review Letters 93(26):268701*, 2004.

[36] J. D. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference, 4nd edition.* Marcel Dekker, 2003.

[37] P. Golle, K. L. Brown, I. Mironov, and M. Lillibridge. Incentives for Sharing in Peer-to-Peer Networks. In *Proc. of the 2nd International Workshop on Electronic Commerce*, 2001.

[38] M. Handley, S. Floyd, J. Pahdye, and J. Widmer. TCP Friendly Rate Control (TFRC) : Protocol Specification. RFC 3448, Jan 2003.

[39] Q. He, C. Dovrolis, and M. Ammar. On the Predictability of Large Transfer TCP Throughput. In *Proc. of ACM SIGCOMM*, August 2005.

[40] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. PROMISE: Peer-to-Peer Media Streaming Using CollectCast. In *Proc. of the 11th ACM international conference on Multimedia*, Berkeley, California, USA, November 2003.

[41] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. Insight into PPLive: Measurement Study of a Large-Scale P2P IPTV System. In *Workshop on Internet Protocol TV (IPTV) Services over World Wide Web, in conjunction with WWW 2006*, May 2006.

[42] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Trans. on Multimedia*, 9(8):1672–1687, December 2007.

[43] X. Hei, Y. Liu, and K. W. Ross. Inferring Network-Wide Quality in P2P Live Streaming Systems. *IEEE Journal on Selected Areas in Communications, Special Issue on Advances in Peer-to-Peer Streaming Systems*, 25(9):1640–1654, December 2007.

[44] R. Janakiraman and L. Xu. Efficient and Flexible Parallel Retrieval using Priority Encoded Transmission. In *Proc. of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2004)*, June 2004.

[45] K. Kar, S. Sarkar, and L. Tassiulas. A Simple Rate Control Algorithm for Maximizing Total User Utility. In *Proc. of IEEE INFOCOM*, April 2001.

[46] N. K. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica 4*, pages 373–395, 1984.

[47] L. G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[48] L. G. Khachiyan. Polynomial Algorithms in Linear Programming. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 20:53–72, 1980.

[49] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat. Using Random Subsets to Build Scalable Network Services. In *Proc. of the USENIX Symposium on Internet Technologies and Systems*, March 2003.

[50] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[51] E. Koutsoupias and C. H. Papadimitriou. Worst-Case Equilibria. *Lecture Notes in Computer Science*, 1563:404–413, 1999.

[52] R. Kumar, Y. Liu, and K. W. Ross. Stochastic Fluid Theory for P2P Streaming Systems. In *Proc. of IEEE INFOCOM*, May 2007.

[53] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for Cooperation in Peer-to-Peer Networks. In *Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[54] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang. Inside the New Coolstreaming: Principles, Measurements and Performance Implications. In *Proc. of IEEE INFOCOM*, April 2008.

[55] B. Li, S. Xie, G. Y. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang. An Empirical Study of the CoolStreaming+ System. *IEEE Journal on Selected Areas in*

Communications, Special Issue on Advances in Peer-to-Peer Streaming Systems, 25(9):1627–1639, December 2007.

[56] J. Li. PeerStreaming: A Practical Receiver-Driven Peer-to-Peer Media Streaming System. Technical report, Microsoft Research MSR-TR-2004-101, September 2004.

[57] L. Li, D. Alderson, W. Willinger, and j. Doyle. A First-Principles Approach to Understanding the Internet's Router-Level Topology. In *Proc. of ACM SIGCOMM*, August 2004.

[58] Z. Li, B. Li, D. Jiang, and L. C. Lau. On Achieving Optimal Throughput with Network Coding. In *Proc. of IEEE INFOCOM*, March 2005.

[59] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante. Characterizing and Predicting TCP throughput on the Wide Area Network. In *Proc. of the 25th International Conference on Distributed Computing Systems (ICDCS 2005)*, June 2005.

[60] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee. Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding. In *Proc. of IEEE INFOCOM*, March 2005.

[61] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks. In *Proc. of ACM SIGMETRICS/Performance'04*, June 2004.

[62] N. Magharei and R. Rejaie. PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming. In *Proc. of IEEE INFOCOM*, May 2007.

[63] M. Mathis, J. Semke, and J. Mahdavi. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review*, 27(3), 1997.

[64] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Boston University Representative Internet Topology Generator. Technical report, http://www.cs.bu.edu/brite, 2000.

[65] D. C. Montgomery, e. A. Peck, and G. G. Vining. *Introduction to Linear Regression Analysis, Third Edition.* John Wiley and Sons, Inc., 2001.

[66] M. E. J. Newman. Assortative Mixing in Networks. *Physical Review Letters 89(208701)*, 2002.

[67] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, 1994.

[68] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. of ACM SIGCOMM*, September 1998.

[69] V. N. Padmanabhan, H. J. Wang, P. A. Chow, and K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. In *Proc. of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002)*, May 2002.

[70] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr. Chainsaw: Eliminating Trees from Overlay Multicast. In *Proc. of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, February 2005.

[71] Y. Qiu and P. Marbach. Bandwidth Allocation in Ad Hoc Networks: A Price-Based Approach. In *Proc. of IEEE INFOCOM*, April 2003.

[72] T. P. Ryan. *Modern Regression Methods.* John Wiley and Sons, Inc., 1997.

[73] H. D. Sherali and G. Choi. Recovery of Primal Solutions when Using Subgradient Optimization Methods to Solve Lagrangian Duals of Linear Programs. *Operations Research Letter*, 19:105–113, 1996.

[74] N. Z. Shor. Cut-off method with space extension in convex programming problems. *Cybernetics 13(1)*, pages 94–96, 1977.

[75] N. Z. Shor. *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985.

[76] B. Sikdar, S. Kalyanaraman, and K. Vastol. Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK. *IEEE/ACM Transactions on Networking*, 11(6):959–971, 2003.

[77] T. Silverston and O. Fourmaux. P2P IPTV Measurement: A Case Study of TVants. In *Proc. of the 2nd Conference on Future Networking Technologies (CoNEXT '06)*, December 2006.

[78] T. Silverston and O. Fourmaux. Measuring P2P IPTV Systems. In *Proc. of the 17th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2007)*, June 2007.

[79] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. In *Proc. of Internet Measurement Conference (IMC)*, October 2005.

[80] G. Tan and S. A. Jarvis. A Payment-based Incentive and Service Differentiation Mechanism for Peer-to-Peer Streaming Broadcast. In *Proc. of the 14th International Workshop on Quality of Service (IWQoS 2006)*, June 2006.

[81] D. A. Tran, K. A. Hua, and T. Do. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In *Proc. of IEEE INFOCOM*, March 2003.

[82] S. Vazhkudai, J. Schopf, and I. Foster. Predicting the Performance of Wide Area Data Transfers. In *Proc. of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*, April 2002.

[83] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt. Measurement and Modeling a Large-Scale Overlay for Multimedia Streaming. In *Proc. of the Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine 2007)*, August 2007.

[84] D. J. Watts. *Six Degrees: the Science of a Connected Age*. ACM Press, 2003.

[85] R. Wolski. Dynamically Forecasting Network Performance Using the Network Weather Service. *Cluster Computing*, 1(1):119–132, 1998.

[86] D. B. Yudin and A. S. Nemirovski. Informational Complexity and Efficient Methods for the Solution of Convex Extremal Problems. *Matekon 13(2)*, pages 3–25, 1976.

[87] M. Zhang, J.-G. Luo, L. Zhao, and S. Yang. A Peer-to-Peer Network for Live Media Streaming - Using a Push-Pull Approach. In *Proc. of ACM Multimedia 2005*, November 2005.

[88] X. Zhang, J. Liu, B. Li, and T. P. Yum. CoolStreaming/DONet: A Data-Driven Overlay Network for Live Media Streaming. In *Proc. of IEEE INFOCOM*, March 2005.

[89] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *Proc. of ACM SIGCOMM*, 2002.