The Impact of Network Address Translation on Peer-to-Peer Live Video Streaming Systems

by

Zhonghua Wei

M.Sc., University of London, 2006

B.Eng., Beijing Univ. of Posts and Telecommunications, 2005

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

The Impact of Network Address Translation on Peer-to-Peer Live Video Streaming Systems

by

Zhonghua Wei

M.Sc., University of London, 2006

B.Eng., Beijing Univ. of Posts and Telecommunications, 2005

Supervisory Committee

Dr. Jianping Pan, Supervisor

(Department of Computer Science)

Dr. Kui Wu, Departmental Member

(Department of Computer Science)

Supervisory Committee

Dr. Jianping Pan, Supervisor

(Department of Computer Science)

Dr. Kui Wu, Departmental Member

(Department of Computer Science)

ABSTRACT

Video streaming over the Internet can be very difficult under the traditional client-server model. Peer-to-peer (P2P) systems, in which each participating peer contributes its upload bandwidth to other peers while it downloads data, have been successful in file-sharing applications, and they appear to be promising in delivering video contents, too. However, the existence of network address translation (NAT) is always considered as a challenge to peer-to-peer systems. NAT has been a practical solution to the Internet Protocol version 4 (IPv4) address exhaustion problem, as it reduces the usage of IP addresses by allowing multiple private hosts to share a single public IP address, but NAT can degrade the performance of a peer-to-peer system as it limits the direction of connectivity. Measurement studies show that a considerable fraction of peer-to-peer video streaming system users are behind NAT devices, and that their uplink bandwidth is not well utilized, but the literature lacks a quantitative analysis of the impact of NAT on the performance of P2P video streaming systems. In this thesis, an extensible analytical model is built to capture the performance for P2P

live streaming systems with a certain percentage of users behind NAT and cannot be reached by NAT traversal techniques, the correctness of which is verified by software simulation. A simple mechanism is proposed in this thesis, which is able to effectively improve the system performance and fairness by counteracting the negative impact of NAT, and it can also be used to reduce the usage of server bandwidth.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr. Jianping Pan, for all his guidance and help during my graduate study. In particular, his full involvement throughout the thesis work ensured that it followed the right path and was successfully completed. The valuable advice from other supervisory committee members, as well as the feedback from fellow graduate students, is much appreciated, too. I am also grateful to my family, especially my wife, for their endless love and support.

DEDICATION

To My Family

Chapter 1

Introduction

Distributing video contents over the Internet to a large number of subscribers is a great challenge, because of the bandwidth-consuming and delay-sensitive nature of video streaming services. Peer-to-peer (P2P) networking is a distributed architecture in which each peer makes their resources, such as network bandwidth, disk storage, and computational power, available to other system participants. File-sharing applications over peer-to-peer systems have been successful, and peer-to-peer systems appear to be quite promising for video streaming applications, too. Compared with the traditional client-server model, in which the server bandwidth will be a bottleneck as the system grows, the peer-to-peer paradigm is much more scalable. Peer-to-peer systems also have the advantage of being cost-effective over IP multicast, which is a technique that depends on the support of network infrastructure. Proposals have been made to enhance BitTorrent [1], a dominant peer-to-peer file-sharing protocol, for video streaming applications. Some other P2P video streaming system designs are inspired by CoolStreaming [2, 3], which is the first real-world system that has attracted thousands of users.

A known problem with peer-to-peer systems is that the existence of network

address translation (NAT) devices can degrade the performance of such systems. NAT [4] is a technique proposed as a solution to the Internet Protocol version 4 (IPv4) address exhaustion problem, which allows multiple network users on a private network to access the Internet by sharing a single public IP address. Nowadays, NAT is widely used because it reduces the usage of IP addresses, simplifies network reconfiguration, and increases the security of private networks. However, NAT violates the end-to-end connectivity across the Internet as it limits the direction of connections by making the hosts behind NAT devices unable to accept incoming connections. Although there are various NAT traversal techniques [5, 6, 7] which make it possible to reach hosts behind NAT devices, they are not always reliable [8, 9, 10]. If NAT traversal is unsuccessful, there is no way for two hosts behind different NAT devices to establish connections with each other.

In the survey papers [11, 12] on system designs, NAT is generally considered as an impediment to high performance peer-to-peer video streaming systems, including both live streaming and Video-on-Demand (VOD) services. NAT impacts the system performance in two ways. First, since the upload bandwidth of the peers behind NAT devices cannot be well utilized, the overall system capacity is reduced, which will be perceived as degraded video performance from the perspective of user experience. To compensate this factor, real-world systems often add many expensive, high-bandwidth servers so as to ensure good user experience by shifting to a hybrid P2P/client-server model. Second, users on the public network can connect to each other as well as receive incoming connections from NAT users, but there are extra restrictions on the connectivity of the latter. The difference in the number of connected peers, or neighbors, results in a performance difference between the two groups, which further leads to a fairness problem. NAT has been identified to have negative impacts on both system capacity and system fairness by measurement studies [13, 14, 15] on

real-world peer-to-peer video streaming systems.

In this thesis, *public peers* are defined as those peers on the public network and can be connected to without restrictions, and *NAT peers* are defined as the ones on private networks behind NAT devices and **cannot** be reached by using NAT traversal techniques. Public peers and NAT peers will be studied under two network scenarios, namely the *bandwidth-constrained* scenario and the *over-provisioned* scenario. In an over-provisioned system, the total upload bandwidth is much higher than that needed to support the video streaming rate (desired average download rate) requirement by all users. On the contrary, a bandwidth-constrained system is one where the total upload bandwidth is barely enough to support the video service.

An analytical model is built for a generic peer-to-peer live streaming system in this thesis, which can be used to capture the system performance in terms of average video continuity index and user share ratio for systems with NAT peers. The model can be used to evaluate the system performance, from the viewpoint of both user experience and system fairness, and to predict the impact of NAT on the system. Both performance metrics are evaluated for public peers and NAT peers separately, so that their performance can be compared with each other. For simulation purposes, a software tool is written to validate the analytical model by cross-checking the simulation data with the numerical results obtained from the analytical model. Two specific simulation scenarios are created, which emulate different broadband Internet access links and target at bandwidth-constrained and over-provisioned systems, respectively. Although the model is built for a generic system, it can be easily extended to model other variants of peer-to-peer video streaming systems. Thanks to the analytical model, it is possible to explore different mechanisms to improve the system performance. A biased peer selection scheme is proposed in this thesis, which is able to effectively alleviate the negative impact of NAT.

The contributions of this thesis are:

(1) To the best of our knowledge, this thesis work is the first study that quantitatively analyzes the impact of NAT on peer-to-peer video streaming systems. The analytical model built in this thesis is capable of precisely predicting the average continuity index and share ratio for a generic P2P streaming system, with the performance metrics being functions of the percentage of total users behind NAT and a set of system parameters. The correctness of the model is supported by software simulation.

(2) The biased peer selection strategy is shown to be an effective performance improvement scheme by both analysis and simulation. The video performance for NAT users and the share ratio for users on the public network are both improved significantly if the biased peer selection scheme is enabled. It also has the potential to reduce the usage of server upload bandwidth. The scheme is easy to implement and only requires minimal modifications to the tracker and client software, and the increased network overhead is negligible since it re-uses existing messages and only adds one parameter which has a constant size. The optimal bias selection factors are determined for systems with different percentage of NAT peers.

(3) The software simulator built as part of this thesis work is able to simulate a large-scale P2P video streaming system. By using the simulator, it is possible to perform extensive performance study for different combinations of system parameters and different network scenarios by modifying the tunable parameters in the configuration file. The common modules of the video streaming systems, such as peer selection strategy, chunk selection strategy, etc., are implemented in a way that they can be easily modified. Other system implementations can be studied with some efforts re-writing individual modules only.

(4) Further discussions on gossip-based peer list dissemination, greedy chunk selec-

tion and heterogeneous bandwidth, as well as alternative performance improvement schemes, are also included in this thesis.

The remainder of the thesis is organized as follows. In Chapter 2, the background information on peer-to-peer video streaming is reviewed, as well as NAT and NAT traversal techniques. The existing research works in the literature that are pertinent to the thesis work are also summarized in this chapter. The step-by-step construction of the analytical model is described in detail in Chapter 3, using which the continuity index and share ratio can be calculated. The software simulator, the simulation scenarios, and the performance evaluation results on the simulated system are presented in Chapter 4. The performance evaluation shows interesting observations on the impact of NAT on user experience and system fairness when some of the system participants are NAT peers. A performance improvement scheme based on a biased peer selection strategy is proposed in Chapter 5, and the improved performance results show the effectiveness of this scheme. Some further discussions on specific topics are included in Chapter 6. Finally, Chapter 7 concludes the thesis and discusses the future work.

Chapter 2

Background and Related Work

In this chapter, the background information relevant to this thesis is reviewed. To be specific, peer-to-peer video streaming systems and network address translation (NAT) are described in general, as well as the correlation between the two. In addition, a summary of the existing research works in the literature is also available in this chapter.

2.1 Video Streaming over the Internet

The Internet has been having huge influences on traditional communications media. Web sites and blogging are taking the place of traditional print publications like newspapers, magazines, and books. Telephone is being replaced by Voice-over-Internet-Protocol (VoIP), supplemented by other personal communication forms such as e-mailing and instant messaging. The distribution of multimedia contents including music, film and television programs is being reshaped by the Internet, too.

However, it is very challenging to distribute real-time multimedia contents over the Internet to a large number of subscribers, especially video contents. Video streaming applications have much higher bandwidth requirements than text, images, and audio

services, and they are delay-sensitive from the perspective of user experience. Because of the Internet's best-effort nature, real-time video streaming over the Internet is a very difficult task unless there is abundant bandwidth for the service.

In the traditional *client-server* model, the server bandwidth will be a bottleneck of a video streaming system, as the size of the system grows. In a system that adopts the client-server model, the server provides the service by answering requests from clients, and clients retrieve information by downloading from the server. As the bandwidth of the server, or a group of servers, is limited, only a limited number of simultaneous subscribers can be supported by the server(s). An increased number of users cannot be served without adding more servers, which is expensive for service providers. In other words, the system is not scalable.

Earlier attempts trying to solve the problem have been made by proposing the *IP multicast* technique. In order to provide a certain level of Quality-of-Service (QoS), IP multicast relies on network routers to manage the data distribution from one source to multiple destinations. Every router maintains the states for each multicast group, and all the routers participating in the same multicast group form a spanning tree to forward packets. Because of its dependency on network infrastructures and the technical complexity, IP multicast is not favored by Internet Service Providers (ISP) and has never been widely deployed.

In recent years, *peer-to-peer* (P2P) systems are increasingly drawing interests from both academia and industry. Unlike in the client-server model, a *participating peer* in a peer-to-peer system downloads not only from the server, but also from other peers. In the meantime, the peer uploads its available data, which it has downloaded to its local storage, to other peers. In that sense, each peer is both a client and a server, as it is both a consumer and a supplier of resources. Peer-to-peer systems are very scalable because each peer contributes its upload bandwidth to other peers in

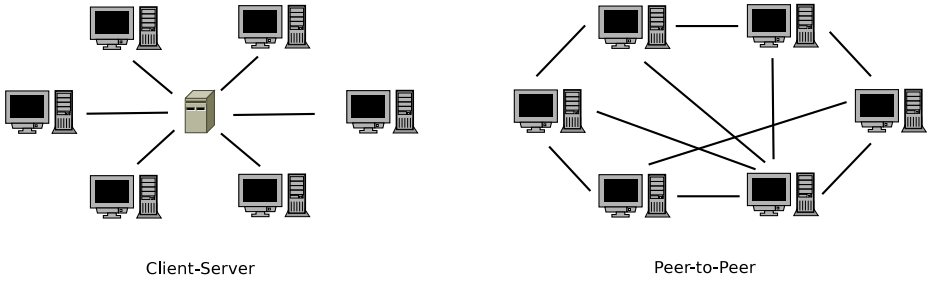

Figure 2.1: Client-Server and Peer-to-Peer Paradigms

the system, so the system has much more resources in addition to that of the server. As more peers join the system, the total upload bandwidth available in the system increases accordingly.

The difference between the client-server paradigm and the peer-to-peer paradigm is illustrated in Figure 2.1. While the server is the sole supplier of resources in the client-server paradigm, a peer-to-peer system does not have such a centralized node.

Peer-to-peer systems have been successful in providing file-sharing services. Popular peer-to-peer file-sharing systems include Gnutella, Napster, and BitTorrent, to name a few. Peer-to-peer systems appear to be promising in delivering video contents, too, but the differences between file-sharing and video streaming must be taken into consideration. With file-sharing, the user does not care much about which part of the file is downloaded first, but the file will not be available to use until it is downloaded in its entirety. In video streaming systems, where the video file is typically divided into smaller pieces, the video pieces must be downloaded in roughly the same order in which they will be played to the user, except within a small window period for scheduling purposes. However, the user can start viewing the program as soon as the beginning part of the video becomes available, and then download the subsequent parts while the video is playing.

2.2 Peer-to-Peer Video Streaming Systems

Recently, academic interests in the design and enhancement of peer-to-peer video streaming system have been continuously growing. Many system proposals have been implemented as prototypes and tested in research environments. In the real world, several commercial systems have been delivering broadcast video contents over peer-to-peer overlay networks, some of which have gained tremendous popularity. This section provides an overview of the current P2P video streaming systems, including both academic and commercial ones.

Several proposals of streaming video contents over peer-to-peer networks are based on BitTorrent [1]. BitTorrent has been a dominant peer-to-peer file-sharing protocol, thanks to its very successful built-in mechanisms. Such mechanisms include a centralized tracker which keeps track of the basic information about all peers downloading in a torrent, the rarest-first piece selection strategy that guarantees the diversity of file pieces in the system, and the tit-for-tat strategy as an incentive mechanism to encourage peers to upload to others. Although BitTorrent was designed for delivering time-insensitive contents, proposals have been made to adopt BitTorrent for video streaming services. BiToS [16] is a system that enhances BitTorrent to distribute video contents by changing the order in which the video file is downloaded, while introducing minimal modifications to the original BitTorrent mechanisms. BASS [17] introduces a hybrid server/P2P streaming system for Video-on-Demand (VOD) services, where clients share pieces using BitTorrent while they download from the streaming server.

CoolStreaming [2] is the first successful peer-to-peer video streaming system that has attracted thousands of users, and most of the current commercial peer-to-peer streaming systems are inspired by CoolStreaming. The key modules in CoolStreaming include a membership manager which maintains a partial view of all participating peers, a partnership manager that decides on which peers to exchange buffer map

and video data with, and a scheduler which manages video data transmission. CoolStreaming is different from the systems based on BitTorrent in several ways: CoolStreaming uses a gossiping protocol [18], instead of a tracker, to disseminate the peer list, and it uses a heuristic scheduling algorithm to make the decision on which particular video chunk to download. However, in the most recent design document of CoolStreaming [3], it has shifted from the originally proposed gossiping membership protocol to the centralized tracker approach. Due to this change, there is no significant difference between CoolStreaming-inspired systems and BitTorrent-based video streaming systems, in terms of peer list dissemination.

Enormous research efforts have been made in attempt to improve P2P video streaming systems. PRIME [19] is a scalable mesh-based P2P streaming mechanism for live content, whose main design goal is to minimize bandwidth bottleneck and content bottleneck. Anysee [20] adopts an inter-overlay optimization scheme for efficient and scalable live-streaming overlay construction. Chainsaw [21] is a P2P overlay multicast system that completely eliminates trees, in which peers are notified of new packets by their neighbors and peers explicitly request packets from a neighbor. Bullet [22] uses a scalable and distributed algorithm that enables nodes spread across the Internet to self-organize into a high bandwidth overlay mesh. SplitStream [23] strips the content across a forest of interior-node-disjoint multicast trees that distributes the forwarding load among all participating peers.

In the business world, PPLive [24], also known as PPTV, has become a leading P2P online video content provider, offering TV shows, movies, news, drama, sports and entertainment programs. PPStream [25] is another popular P2P streaming system that provides a wide variety of programs including Chinese movies and Korean drama, as well as popular American TV shows and films. UUSee [26] is a large-scale live and on-demand streaming system, in which random network coding has

been first used as the cornerstone in production deployment. While these successful peer-to-peer video streaming systems, along with other existing commercial systems such as SopCast and TVAnts, offer medium-to-low-quality narrow-band P2P video streaming, a new generation of commercial P2P streaming applications is being deployed [27], including Zattoo [14], Joost, and Babelgum, and they will be offering high-quality video services over P2P networks.

2.3 Network Address Translation (NAT)

Network address translation, or NAT, has been a practical solution to the Internet Protocol version 4 (IPv4) address exhaustion problem. The reason is that NAT allows multiple hosts on the same private network to access the Internet by sharing a single public IP address. NAT devices, often NAT-enabled routers, have now become a common, indispensable component in most home or small-office network settings.

NAT is sometimes referred to by other names [28], such as network address and port translation (NAPT), port address translation (PAT), NAT overload and IP masquerading. NAT behaviors can be classified as *full-cone NAT* (one-to-one NAT), *(address) restricted cone NAT*, *port-restricted cone NAT* and *symmetric NAT* [5], and many real-world NAT implementations are combinations of these types.

In general, NAT works by mapping the private address (10.x.x.x, 172.16.x.x through 172.31.x.x, and 192.168.x.x) of an internal host to the public address assigned by an Internet Service Provider (ISP). This typically involves modifications to IP address, IP header checksum, and any higher level checksums that include the IP address in the checksum calculation. If there are multiple hosts behind a NAT device, higher layer control information in the packets, such as Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) port numbers, is also altered, and

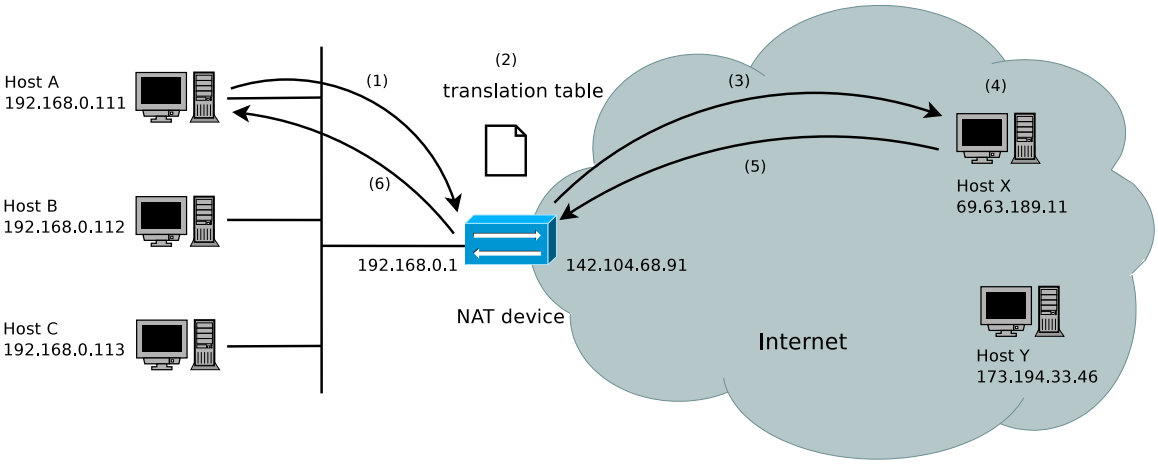

Figure 2.2: Network Address Translation

a translation table is maintained by the NAT device. TCP/UDP port numbers are then utilized to de-multiplex return packets to the internal hosts, in order to ensure that addresses can be properly translated back and that packets are delivered to the correct host.

Figure 2.2 illustrates a scenario how multiple hosts behind a NAT device share a single public IP address, and how a connection is established between an internal host on the private network and an external host on the public network.

In this example, the two-way connection between Host A and Host X is established in the following steps:

(1) Host A on the private network sends a packet to external Host X (src = 192.168.0.111:5080, dest = 69.63.189.11:80), and the packet is first routed to the internal interface of the NAT device.

(2) NAT selects a port number from a pool of available ports, and assigns it to the connection. NAT then creates a new entry in its translation table, mapping the chosen port number to the internal IP address and original source port combination (8000 <=> 192.168.0.111:5080). All subsequent packets for this particular connection are translated to the same port number.

(3) NAT replaces the internal IP address in the source field of the IP header with the external IP address of the NAT device, and the original port number in the source port field with the port number assigned to the connection from Step 2 (src = 142.104.68.91:8000, dest = 69.63.189.11:80). The altered packet is then sent to the public network.

(4) Host X receives the packet that has been altered by NAT and establishes a connection to the IP address and the port number specified in the received packet header, oblivious to the fact that they have been translated.

(5) Host X sends a return packet to Host A, but the packet is sent to the public IP address of NAT (src = 69.63.189.11:80, dest = 142.104.68.91:8000).

(6) The return packet coming from the public network is mapped by NAT to a corresponding internal IP address and port number by looking up the translation table, and NAT uses them to replace the destination IP address and port number in the packet header before forwarding the packet to the internal network (src = 69.63.189.11:80, dest = 192.168.0.111:5080).

Note that in this particular example, it is the internal host that initiates the connection. However, it is impossible for the external host X to initiate a connection to an internal host, say Host B, if they have not communicated with each other before and NAT does not have a corresponding mapping in the translation table. If NAT receives a packet whose destination port number cannot be found in the translation table, the packet is rejected or discarded because the NAT device does not know which internal host to send the packet to. This effectively breaks the end-to-end connectivity in the originally envisioned Internet model, and it will be further discussed in the following sections.

2.4 NAT Traversal

The previous section has shown that hosts behind NAT devices cannot receive incoming connection requests, unless there is an existing entry in the translation table maintained by the NAT device. As a result, it is difficult if two hosts behind distinct NAT devices try to communicate with each other. One solution is to use *port forwarding*, which involves static configuration of translation table entries by network administrator. Another solution is to use NAT traversal techniques to establish and maintain network connections, *traversing* NAT devices. Many of NAT traversal techniques require a publicly routable server, which either assists with establishing the connection or relays all packets. Some popular NAT traversal techniques are briefly summarized below.

Session Traversal Utilities for NAT, or STUN, formerly known as Simple Traversal of UDP over NATs (STUN) protocol, allows applications to discover the presence of NAT and to obtain the mapping that the NAT has allocated for the application's User Datagram Protocol (UDP) connections to remote hosts. STUN requires the assistance from a third-party server on the public network. Traversal Using Relay NAT (TURN) is an extension to STUN. TURN allows a host behind a NAT device (TURN client) to request another host (TURN server) to act as a relay. Interactive Connectivity Establishment (ICE) makes use of STUN and TURN, and it can be used by any protocol utilizing the offer/answer model, such as the Session Initiation Protocol (SIP). Besides these techniques based on NAT behaviors, protocols such as Internet Gateway Device (IGD) Protocol and Middlebox Communications (MIDCOM) Protocol allow applications running on end hosts to explicitly control the NAT device to facilitate connections. However, this requires that some special protocols like Universal Plug and Play (UPnP) to be available and enabled on the NAT device.

In general, NAT traversal for Transmission Control Protocol (TCP) connections is

more challenging than for UDP connections, because of the asymmetric nature of TCP connection establishment. Studies such as NUTSS [29] and NATBlaster [30], have been successful in establishing TCP connections passing through NAT. The success of these approaches depends on how NAT devices respond to various sequences of TCP and Internet Control Message Protocol (ICMP) packets. Measurement work in [9] is a comprehensive study on various TCP NAT traversal techniques over a wide range of commercial NAT products, and a test suite is developed to estimate the likelihood of successful NAT traversal for home networks.

Although many NAT traversal techniques exist, no single method works in every situation, since NAT is implemented via a number of different address and port mapping schemes, none of which are standardized until recent standardization efforts such as [31, 32]. Studies [8] on a simple but practical NAT traversal technique, commonly known as "hole punching", suggest that the success rate of hole punching is about 82% for UDP and about 64% for TCP. The comprehensive experiments in [9] report better results in successfully traversing NAT for TCP (88% on average), but a failure rate of 11% is still not acceptable to many applications. Studies in [10] show that although the success rate of NAT traversal techniques can be up to 95.14%, it can also be as low as 17.30%, depending on the Service Categories (Requester side NAT Traversal, Global Service Provisioning, Service Provisioning using Pre-Signaling, and Secure Service Provisioning) defined in their paper.

If NAT traversal is not successful, a host behind NAT has difficulties receiving incoming connection requests, and the host can only be the client, but not the server, in the client-server paradigm. This poses serious problems to peer-to-peer systems, where each host is expected to play both the client role and the server role.

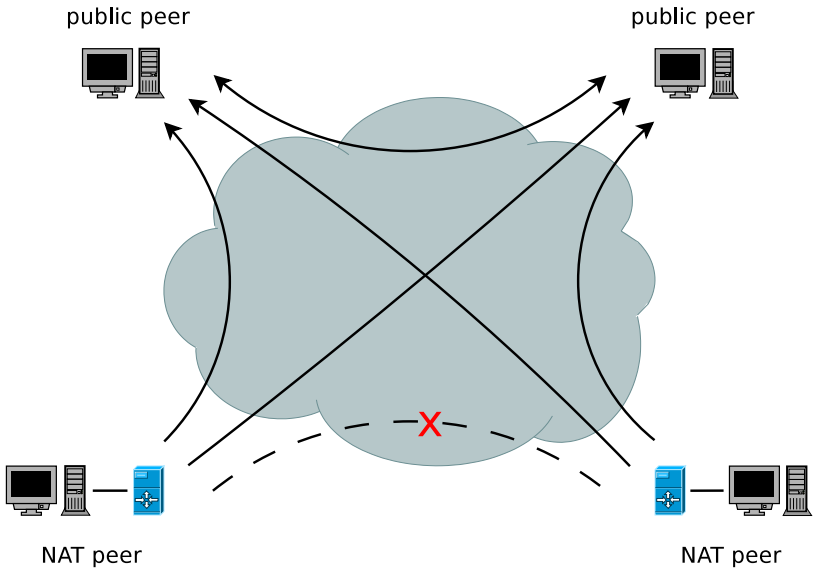

Figure 2.3: NAT in Peer-to-Peer Systems

2.5 NAT and Peer-to-Peer Applications

The existence of NAT is a known problem for essentially any peer-to-peer systems. The root cause is that NAT limits the direction of connectivity by making hosts behind NAT devices unable to accept incoming connections, as shown in the previous sections. Unless NAT traversal (which is not reliable) is successful, there is no way for two hosts behind different NAT devices to communicate between each other, since no matter which host initiates the connection, the connection request is denied by the NAT on the remote side.

Figure 2.3 illustrates how NAT impacts a peer-to-peer system. The arcs show all possible connections between hosts in this simple network, and the arrows indicate the direction of the connections, i.e., which host initiates the connection. Two hosts both on the public network can communicate with each other without any restrictions. Assuming NAT traversal is not successful, a host behind NAT can only communicate with a host on the public network if the former is the one that initiates the connection, but two hosts behind different NAT devices cannot establish connections with each

other, no matter which host tries to initiate the connection.

Since a participating peer in a P2P system heavily relies on the contributions from other peers, in a bandwidth-constrained system, a smaller number of connected peers will results in worse performance, especially for the peers behind NAT. Even in an over-provisioned system, the difference in neighborhood sizes and video performance between peers behind NAT devices and those on the public network leads to a fairness issue across the peer-to-peer system.

NAT is generally considered as a negative factor in the design of peer-to-peer systems, including peer-to-peer video streaming systems. In [11], the authors consider NAT as a challenge when designing a P2P video broadcast system, and the authors of [12] list NAT (and firewalls for similar reasons) as an impediment to P2P Video-on-Demand (VOD) systems, too. As a matter of fact, measurement studies [15, 33] show that up to over a half of the users of real-world peer-to-peer video streaming systems are connected to the Internet via NAT devices. According to the measurement presented in [34], the percentage of users that are not on the public network (combining NAT and firewalls together) in North America and Europe is close to 90% (the success rate of NAT traversal is not available in this work, but it can be estimated by [8, 9, 10]). The strong presence of NAT in peer-to-peer systems will have a significant impact on the system performance.

2.6 Related Work

This section summarizes the existing work related to this thesis in the literature, and discusses how this thesis is unique from others.

The performance of peer-to-peer systems, video streaming systems in particular, has been intensively studied by means of measurement, analytical modeling, simu-

lation and emulation, and the combinations of these methodologies. Qiu *et al.* [35] built a fluid model that can be used to study the scalability, performance and efficiency of BitTorrent-like peer-to-peer systems. The built-in incentive mechanism of BitTorrent was considered and its effect on system performance was studied. Tewari *et al.* [36] provided an analytical model for BitTorrent-based live video streaming systems, focusing on the efficiency of such systems. Their argument was that a pure P2P solution can only support limited streaming rates, but adding a well-designed P2P solution to existing server-based streaming systems is able to achieve substantially higher streaming rates. Zhou *et al.* [37] created a buffer model to study and compare different data-driven downloading strategies, and the performance metrics used are continuity index and startup delay. Both rarest-first strategy and greedy strategy were explored and a mixed strategy between the two was proposed. Kumar *et al.* [38] developed a stochastic fluid model for peer-to-peer streaming systems, accounting for many of the essential features of such systems. Their model can be used to provide closed-form expressions in order to understand the fundamental behavior of P2P streaming systems. Liu *et al.* [39] gave performance bounds, in terms of the server capacity, video quality, and depth of the distribution trees, for peer-assisted live streaming. They also presented algorithms for constructing the trees that achieve such performance bounds. Wang *et al.* conducted measurement studies [40] to understand the potentials and challenges of providing streaming service, and their results showed that most peer sets in BitTorrent-based video file swarms are good enough to support streaming services.

NAT in peer-to-peer systems has also attracted some research interests. According to the measurement studies in [33], more than 50% of PPLive peers are NAT users and do not respond to contact initiated from the outside. Measurement on the CoolStreaming system [15] showed some similar results that up to 65% of the peers

are behind NAT, and they do not contribute as much as their counterparts on public networks. Authors of [13] have shown that the upload capacity of a node behind NAT is not properly utilized and a high capacity node is considered bad from the viewpoint of the application if it is attached to a NAT. Authors of [14] have reported reduced capacity in their measurement study, too, no matter which one of the four types of NAT device is used. Mol *et al.* [41] showed that there is a fairness issue when a large fraction of the peers are behind firewalls (which applies to NAT devices, too). Research efforts have also been made to perfect NAT traversal techniques in the context of peer-to-peer systems, such as [42] and [43].

These research works are all related to the subject of peer-to-peer systems, but each of them has its own research interests and focuses on a different aspect. Although the existing works cannot be used to directly solve our problem, they can give this thesis inspirations, e.g., a part of our analytical model is inspired by Zhou's work [37]. However, their work is most interested in evaluating the chunk scheduling algorithms, but does not consider the existence of NAT at all. Furthermore, they do not study system fairness, or attempt to improve system performance by counter-acting NAT's negative impacts. Similarly, none of the other existing works are able to quantitatively analyze NAT's impact on the peer-to-peer video streaming systems.

The preceding research work of this thesis [44, 45] was able to accurately estimate the impact of NAT on peer-to-peer file-sharing systems (D'Acunto *et al.* have worked on a very similar topic [46]). In fact, this thesis topic was suggested as a follow-on research topic by the preceding work. However, despite some similarities in terms of research approach, this thesis is unique from the existing work. First, file-sharing systems and video streaming systems are distinct in their nature, and there are a lot of differences in analyzing these systems, as different performance metrics make more sense for each system: While a user of a file-sharing application only cares about

the download time, whether a viewer is able to continuously watch the video reflects the user experience much better in video streaming systems. Second, the analytical model in this thesis is built in a distinct way from the previous work, with only a small portion re-using the existing results. The re-used results only appear in one of the several modules (discussed in more details in Chapter 3) in the modeling work of this thesis, and all the other modules have little dependency on it. Also, a brand new software simulator is written from scratch for simulation purposes, without re-using any of the existing software. Third, this thesis includes some further discussions on various related topics, such as the gossip-based membership protocol and reducing the server bandwidth usage, which are not available in the previous work. Last but not least, some of the performance evaluation results for P2P video streaming systems in this thesis work are surprisingly different from those for P2P file-sharing systems in the previous work, for instance the fairness results on share ratio, which will be discussed in more details in later chapters. Therefore, the research work presented here is distinct from existing studies, and has enough contributions to be an independent thesis topic.

2.7 Summary

The pertinent background information is provided in this chapter. Peer-to-peer systems appear to be promising in delivering video contents over the Internet, as the peer-to-peer paradigm is more scalable than the traditional client-server model, and it is more cost-effective than IP multicast. NAT has been widely used in various network settings as a practical solution to the IPv4 address exhaustion problem, but at the cost of violating the end-to-end connectivity across the Internet and limiting the direction of connections. Many NAT traversal techniques exist, but they can-

not be fully relied on in reaching the hosts behind NAT devices due to the lack of standardization in NAT manufacturing in the past. NAT is generally considered as an impediment in peer-to-peer system designs, because it prevents NAT peers to contribute their uplink bandwidth, as suggested by measurement studies, which deteriorates the system performance. In the literature, there has been various modeling works targeting at different aspects of peer-to-peer video streaming systems, but so far none of them are capable of quantitatively analyzing the impact of NAT on such systems.

Chapter 3

Analytical Model

In order to quantitatively understand the impact of NAT on peer-to-peer video streaming systems, an analytical model is built to capture the system performance with a certain percentage of NAT users. The performance metrics are video continuity index and share ratio, which are used to reflect user experience and system fairness, respectively. Define *public peers* as the peers that are in the public network and can be reached without any restrictions, and *NAT peers* the peers that are behind NAT devices and **cannot** receive incoming connection requests, i.e., NAT traversal is **not** successful. In particular, it is assumed in this thesis that there is only one host behind a NAT device using the peer-to-peer video service at a time. Although it is a common practice that there are multiple hosts behind a NAT device, this is treated as one host being a proxy for others, and we are only interested in the proxy host, which is considered as one single NAT peer. By using the analytical model, both continuity index and share ratio can be derived as mathematical functions of the percentage of NAT peers in the system and a set of system parameters.

This chapter describes how the analytical model is built for a generic peer-to-peer video streaming system in detail. The generic system, as well as the necessary

Table 3.1: List of System Parameters

α	the percentage of the NAT peers in the system
N	the total number of the peers in the system
K	the number of peers returned by the tracker
U_s	server upload capacity
U	peer upload capacity
D	peer download capacity
B	the length of the video streaming buffer

assumptions and simplifications, is introduced first, followed by the step-by-step construction of the model. The model is cross-checked and validated by simulation results presented in Chapter 4. Note that although the model is initially built for a generic system, it can be easily modified and adapted to other variants of peer-to-peer video streaming systems, e.g., with alternative peer selection strategy, alternative chunk selection strategy, different network scenarios, etc. How the model can be extended is shown later in Section 6.1.

For smooth reading experience and a better understanding of the contents in this chapter, Table 3.1 lists all the system parameters and a list of mathematical symbols is available in Table 3.2. The parameters and symbols are also explained in individual sections of this chapter where they are first used.

3.1 System Model

Like any analytical modeling work, it is necessary to make some assumptions and simplifications on the real systems so that some of their key features can be captured and analyzed, with minor issues ignored. This modeling work is initially targeted at a generic peer-to-peer video streaming system, which is described in this section in terms of peer list dissemination method, peer selection strategy, chunk selection strategy, video buffering, etc. Only live streaming is analyzed but not Video-on-

Table 3.2: List of Symbols

N_p	the total number of public peers
N_n	the total number of NAT peers
L_{pp}	the number of public neighbors of a public peer
L_{pn}	the number of NAT neighbors of a public peer
L_p	the total number of neighbors of a public peer
L_n	the total number of neighbors of a NAT peer
ρ_{pp}	the probability that a public peer selects a public neighbor
ρ_{pn}	the probability that a public peer selects a NAT neighbor
ρ_n	the probability that a NAT peer selects a (public) neighbor
Q_p	the average number of download requests sent to a public peer
Q_n	the average number of download requests sent to a NAT peer
$P_p(i)$	the probability that a public peer receives i requests
$P_n(i)$	the probability that a NAT peer receives i requests
R_p	the expected number of responses sent by a public peer
R_n	the expected number of responses sent by a NAT peer
Φ_p	the response probability of public peers
Φ_n	the response probability of NAT peers
$S_{pp}(i)$	the prob. that a public peer requests piece i from a public neighbor
$S_{pn}(i)$	the prob. that a public peer requests piece i from a NAT neighbor
$S_{np}(i)$	the prob. that a NAT peer requests piece i from a (public) neighbor
$G_{pp}(i)$	the prob. that a public peer gets piece i from a public neighbor
$G_{pn}(i)$	the prob. that a public peer gets piece i from a NAT neighbor
$G_p(i)$	the probability that a public peer gets piece i from a neighbor
$G_n(i)$	the probability that a NAT peer gets piece i from a (public) neighbor
$H_p(i)$	the probability that a public peer has piece i in its streaming buffer
$H_n(i)$	the probability that a NAT peer has piece i in its streaming buffer
CI_p	the continuity index of public peers
CI_n	the continuity index of NAT peers
D'_p	the number of raw downloads of a public peer
D'_n	the number of raw downloads of a NAT peer
U'_p	the number of raw uploads of a public peer
U'_n	the number of raw uploads of a NAT peer
SR_p	the average share ratio of public peers
SR_n	the average share ratio of NAT peers

Demand (VOD) services, as VOD systems can be more difficult to model because of the lack of synchronization between peers. The impact of NAT on peer-to-peer VOD systems can be a follow-on research work of this thesis.

The system model makes the following assumptions and simplifications which are similar to other analytical work. Possible alternatives for each component are also briefly introduced here, some of which will be further discussed in Chapter 6.

(1) There is a centralized *tracker* in the system. When a *new peer* joins the system, it contacts the tracker first and the tracker returns a *partial participant list* containing the information of up to K peers. The K peers are randomly chosen from all peers alive in the system. The new peer contacts each of the peers on the partial list and initiates and establishes a connection if the other party is not a NAT peer. The tracker-based peer list dissemination method is essentially the same as a BitTorrent file-sharing system [1]. A possible alternative is to use a gossip-based approach, like that adopted in the original CoolStreaming system [2, 18], which is discussed in Section 6.2.

(2) After the bootstrapping stage, the peer stays in the system until the end of the video, which is the expected behavior of a user watching a popular program. A peer does not ask the tracker for an updated peer list, or use Peer Exchange (PEX) to exchange its list with other peers, after the initial contact with the tracker, i.e., the neighborhood remains *static* afterwards in the entire peer lifetime. This is a reasonable assumption as peer list updating is most useful in handling network dynamics, e.g., frequent peer joining and leaving, which is excluded from the model. Enabling peer list updating in our model can refresh each peer's neighborhood periodically. The neighborhood size can also be increased, but as long as the tracker does not treat public peers and NAT peers differently, both the neighborhoods of public and NAT peers are expanded in a similar way, which leads to no significant difference on system

performance (related discussions on using a larger K value as the system parameter are available in Section 6.4).

(3) Each peer has a *streaming buffer* of the same length B, which is essentially a sliding window. Each slot in the buffer can hold a *video chunk*. All buffers of all peers are assumed to be *synchronized*, i.e., time is slotted and peers always watch the same part of video. Similar assumptions on the streaming buffer are also made in [37].

(4) There is one *video server* distributing the video content, and the video content is chopped into chunks, each containing one time slot of the video. The upload capacity of the server is U_s, which means that the server *pushes* one video chunk to each of the U_s *uniformly at random* selected peers in one time slot. Assuming the video server is publicly known, all peers (both public and NAT) have an equal chance to download from the server. *Random peer selection* strategy is also used in the communication between peers. Research [47] on epidemic live streaming (where epidemic-style is defined as "the stream is divided into small parts, so-called chunks, that follow random, independent paths in the peer population") shows that the random peer selection strategy is part of the optimal combination of the mechanisms that achieve the optimal video dissemination rate. Some real systems such as CoolStreaming [2, 3] rank peers based on a scoring mechanism, and a peer always selects its neighbors with better scores (the score is based on a peer's connection speed and data availability, which essentially makes the peer selection process non-random). The scoring mechanism can effectively help a peer find the fastest connections among its neighbors, but this is less desirable in our system where peer bandwidth is assumed to be homogeneous (see below). A *biased peer selection strategy* is explored as a performance improvement scheme in Chapter 5.

(5) In each time slot, a peer randomly chooses D of its neighbors and sends one download request to each of them, so that up to D chunks can be downloaded in a

single time slot. In other words, the *download capacity* of each peer is D. *Rarest-first* strategy is used for chunk selection, just as that in BitTorrent file-sharing systems, which means when a peer requests for missing chunks in its own buffer that can be downloaded from its neighbors, it always selects the chunk whose availability is the lowest in its neighbors. Since the server disseminates chunks sequentially, it can be assumed that such a chunk is always the one furthest away from the playback point, which is the rarest-first approximation used in [37], too. An ideal scenario is assumed where a peer always knows the most up-to-date buffer bitmap of its neighbors. Note that it is possible that a peer asks for different video chunks in the D download requests, depending on its bitmap information and that of the chosen neighbors. The authors of [16] show that in video streaming systems, the diversity of pieces and high QoS can be guaranteed by using the rarest-first strategy within a *moving window* of the video file. The rarest-first philosophy is used in real systems. For instance, the scheduler implementation in CoolStreaming identifies the video chunks with fewer potential suppliers and tries to download them first. Other chunk selection strategies include *greedy strategy*, where higher priorities are given to chunks that are closest to the playback point, and mixed strategy combining both rarest-first and greedy strategies. Further discussions on the greedy chunk selection strategy are included in Section 6.1.1.

(6) A peer can respond to up to U download requests in a time slot, or the *upload capacity* is U. If more than U requests are received, it *randomly* chooses U of them to respond so that the maximum number of chunks a peer can upload in any time slot is U, and the rest of the requests are simply discarded. Unlike the BitTorrent system, no *tit-for-tat* strategy is implemented for video streaming. In [12], the authors argue that tit-for-tat fails to work as an incentive to induce peers to help each other in P2P streaming systems. The authors of [17] argue that in P2P live streaming systems,

there is no incentive for peers to contribute their upload capacity other than good faith.

(7) The upload capacity and download capacity for all peers are assumed to be the same, i.e., a *homogeneous bandwidth* scenario is assumed. Section 6.1.2 includes some further discussions on a *heterogeneous bandwidth* scenario.

Some of the above-listed items are design choices, and a real system can choose one strategy or another for a particular module (such as peer selection and chunk selection). One system can be closer to ours than another, but an exhaustive study on all strategy combinations is a heavy task with limited values. Some other items listed here are necessary to simplify the modeling by ignoring less critical factors and focusing on the major problem. For example, although every peer is likely to have a different bandwidth condition in reality, a very difficult (if not impossible) modeling task is greatly simplified by assuming the peer bandwidth to be homogeneous. Even with these assumptions and simplifications, our analytical model can still give insights on realistic systems. First, the fundamental reasons behind NAT's impact on system performance are always the same, not matter what the design choices are. Second, our model can be easily extended (discussed in Section 6.1) by re-modeling its individual modules, so that it can quantitatively analyze other system variants than the generic system described above.

Based on the assumptions and simplifications made above, the analytical model can now be built to evaluate the performance of peer-to-peer live video streaming systems, in the presence of NAT devices.

3.2 Number of Neighbors

Previous work [44, 45] has shown that in a BitTorrent system, NAT peers have fewer neighbors than public peers on average. Since we are analyzing peer-to-peer systems which use a tracker-based approach to disseminate the peer list, the same argument for the peer neighborhood sizes can apply here (verified by simulation in the next chapter).

The total number of public peers N_p, and the total number of NAT peers N_n can be easily determined as:

$$N_p = (1 - \alpha)N \tag{3.1}$$

$$N_n = \alpha N \tag{3.2}$$

where N is the total number of peers in the system and α is the percentage of NAT peers.

Let the number of peers contained in a tracker response be K. Denote by L_{pp} the number of public neighbors of a public peer, L_{pn} the number of NAT neighbors of a public peer, and L_p the total number of neighbors of a public peer. Denote by L_n the total number of (public) neighbors of a NAT peer. We have [44, 45]:

$$L_{pp} \approx 2(1 - \alpha)K \tag{3.3}$$

$$L_{pn} = (1 - \alpha)K \cdot \frac{N_n}{N_p} = \alpha K \tag{3.4}$$

$$L_p = L_{pn} + L_{pp} = (2 - \alpha)K \tag{3.5}$$

$$L_n = (1 - \alpha)K \tag{3.6}$$

3.3 Response Probabilities

In each time slot, it is possible that peers cannot respond to all download requests they have received because of the limit of their upload capacity U. If a peer receives more than U download requests in one time slot, some of them will be dropped. Public peers and NAT peers are likely to have different request response probabilities due to the difference in their neighborhood sizes.

In order to model the request response probabilities, we need to calculate the average number of download requests that a peer receives in a time slot, and the expected number of responses it can grant.

3.3.1 Average Number of Requests Received

Denote by ρ_{pp} the probability that a public peer selects a public neighbor to send a download request, and ρ_{pn} the probability that a public peer selects a NAT neighbor. Denote by ρ_n the probability that a NAT peer selects a (public) neighbor to request for video chunks. Because a random peer selection strategy is used, the probabilities are simply:

$$\rho_{pp} = \rho_{pn} = \frac{1}{L_p} \tag{3.7}$$

$$\rho_n = \frac{1}{L_n} \tag{3.8}$$

In each time slot, all public peers send $D \cdot (\rho_{pp} L_{pp}) \cdot N_p$ download requests to all public peers, and $D \cdot (\rho_{pn} L_{pn}) \cdot N_p$ requests to all NAT peers. All NAT peers send $D \cdot (\rho_n L_n) \cdot N_n = D \cdot N_n$ requests to all public peers in a time slot. Assuming steady state and due to symmetry, the average number of download requests a public or NAT peer receives (Q_p or Q_n) should be the total number of requests sent to all public or

NAT peers divided by N_p or N_n, i.e.,

$$Q_p = \frac{D \cdot ((\rho_{pp} L_{pp}) N_p + N_n)}{N_p} \tag{3.9}$$

$$Q_n = \frac{D \cdot (\rho_{pn} L_{pn}) N_p}{N_n} \tag{3.10}$$

3.3.2 Expected Number of Responses

In each time slot, a peer randomly chooses D neighbors and sends a download request to each of the chosen neighbors. The probability that a public peer chooses a particular neighbor to send a request is $\binom{L_p-1}{D-1}/\binom{L_p}{D} = D/L_p$, and the probability is $\binom{L_n-1}{D-1}/\binom{L_n}{D} = D/L_N$ for NAT peers.

$P_p(i)$, the probability that a public peer receives i download requests in a time slot, is the sum of the probabilities that it receives j requests from other public peers (j public neighbors out of L_{pp} send requests to it but not the other $L_{pp} - j$ public neighbors) and $i - j$ requests from NAT peers ($i - j$ NAT neighbors out of L_{pn} send requests to it but not the other $L_{pn} - (i - j)$ NAT neighbors), where $0 \leq i \leq L_P$ and $0 \leq j \leq i$, i.e.,

$$P_p(i) = \sum_{j=0}^{i} \binom{L_{pp}}{j} (\rho_{pp} D)^j (1-\rho_{pp} D)^{L_{pp}-j} \cdot \binom{L_{pn}}{i-j} (\rho_n D)^{i-j} (1-\rho_n D)^{L_{pn}-(i-j)} \tag{3.11}$$

$P_n(i)$, the probability that a NAT peer receives i download requests in a time slot, is the probability that i of its (public) neighbors out of L_n send their requests but not the other $L_n - i$ neighbors, where $0 \leq i \leq L_N$, i.e.,

$$P_n(i) = \binom{L_n}{i} (\rho_{pn} D)^i (1 - \rho_{pn} D)^{L_n - i} \tag{3.12}$$

When the total number of requests i is greater than U, the peer randomly chooses

U of these requests to respond and discard all the others, otherwise it responds to all of the i received requests. The number of uploads by a public peer in a time slot R_p and the number of uploads by a NAT peer R_n can be calculated using the following equations:

$$R_p = \sum_{i=1}^{U}(i \cdot P_p(i)) + \sum_{i=U+1}^{L_P}(U \cdot P_p(i)) \tag{3.13}$$

$$R_n = \sum_{i=1}^{U}(i \cdot P_n(i)) + \sum_{i=U+1}^{L_N}(U \cdot P_n(i)) \tag{3.14}$$

Finally, the probability that a public or NAT peer responds to a download request, Φ_P or Φ_N, can be calculated as follows:

$$\Phi_p = \frac{R_p}{Q_p} \tag{3.15}$$

$$\Phi_n = \frac{R_n}{Q_n} \tag{3.16}$$

Under this model, given fixed values of U and D, public peers respond to download requests at a lower probability than NAT peers due to their larger neighborhood and hence the contention between neighbors. Since NAT can only download from public peers while public peers can download from both public and NAT peers, this fact indicates that public peers have better chance to download a useful piece from their neighbors than NAT peers.

3.4 Streaming Buffer Model

On top of Zhou *et al.*'s buffer model [37], peers are divided into two categories (public and NAT), and the average number of neighbors, as well as the response probabilities derived above, are taken into consideration in this thesis work, as follows.

The buffer of each peer is modeled as a sliding window of B chunks (illustrated

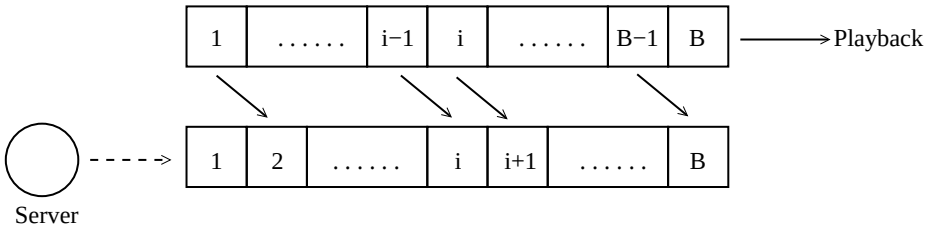

Figure 3.1: Streaming Buffer Model

in Figure 3.1). In each time slot, the oldest chunk (i.e., the B-th chunk) is played (if available) and removed from the buffer, and all the other chunks in the buffer are shifted by one slot towards the playback point. Therefore, the probability that a peer has chunk $i+1$ in its buffer is the sum of the probability that it already has chunk i before the shifting and the probability that it downloads chunk i to its buffer in the time slot, which is a recursive process. Denote by $H_p(i)$ and $H_n(i)$ the probability that a peer has chunk i in the buffer, for public peers and NAT peers, respectively.

3.4.1 Chunk Selection

Because the rarest-first strategy is used for chunk selection, the conditions that a peer requests a particular chunk i to download from its neighbor are:

(a) Chunk i is missing from the buffer of the requesting peer but is available in the requested peer's buffer; and

(b) For chunks 1 through $i-1$, either the peer already has it in the buffer, or neither the peer itself nor the requested peer has it in the buffer.

Based on such conditions, the following equations calculate $S_{pp}(i)$, the probability that a public peer selects chunk i to download from a public neighbor, $S_{pn}(i)$, the probability that a public peer selects chunk i to download from a NAT neighbor, and $S_{np}(i)$, the probability that a NAT peer selects chunk i to download from a public

neighbor,

$$S_{pp}(i) = (1 - H_p(i))H_p(i) \cdot \prod_{j=1}^{i-1}(H_p(j) + (1 - H_p(j))(1 - H_p(j))) \tag{3.17}$$

$$S_{pn}(i) = (1 - H_p(i))H_n(i) \cdot \prod_{j=1}^{i-1}(H_p(j) + (1 - H_p(j))(1 - H_n(j))) \tag{3.18}$$

$$S_{np}(i) = (1 - H_n(i))H_p(i) \cdot \prod_{j=1}^{i-1}(H_n(j) + (1 - H_n(j))(1 - H_p(j))) \tag{3.19}$$

3.4.2 Chunk Availability

Only if a peer selects chunk i to download and the remote peer is responding, can chunk i be download to the buffer. Denote by $G_{pp}(i)$ the probability that a public peer successfully gets chunk i from another public peer, $G_{pn}(i)$ the probability that it downloads, or "gets", the chunk from a NAT peer, and G_p the sum of $G_{pp}(i)$ and $G_{pn}(i)$. Similarly, denote by $G_n(i)$ the probability that a NAT peer gets chunk i from a (public) peer.

$$G_{pp}(i) = S_{pp}(i) \cdot (\rho_{pp} L_{pp})\Phi_p \tag{3.20}$$

$$G_{pn}(i) = S_{pn}(i) \cdot (\rho_{pn} L_{pn})\Phi_n \tag{3.21}$$

$$G_p(i) = G_{pp} + G_{pn} \tag{3.22}$$

$$G_n(i) = S_{np}(i) \cdot \Phi_p \tag{3.23}$$

It is now possible to recursively compute the probability that a peer has chunk $i + 1$ to the buffer after the shift, where $1 \leq i < B$ and B is the buffer size:

$$H_p(i + 1) = H_p(i) + (1 - (1 - G_p(i))^D) \tag{3.24}$$

$$H_n(i+1) = H_n(i) + (1 - (1 - G_n(i))^D) \tag{3.25}$$

where the second term in both equations can be interpreted as "unless none of the D download requests successfully downloads chunk i to the buffer".

The initial condition for the recursions is:

$$H_p(1) = H_n(1) = \frac{U_s}{N} \tag{3.26}$$

because of the random server contribution, where U_s is the upload capacity of the server.

3.5 Continuity Index

The *continuity index* reflects how likely that a user can view the video continuously. It is mathematically defined as the probability that a chunk has been downloaded to the buffer by the time of its playback deadline, or simply the probability of having the B-th chunk, which is at its playback deadline, in the video streaming buffer:

$$CI_p = H_p(B) \tag{3.27}$$

$$CI_n = H_n(B) \tag{3.28}$$

3.6 Share Ratio

Peer-to-peer systems are designed to be fair and *share ratio* is often used as a metric to reflect of a system's fairness. Share ratio is defined as the total number of video chunks that a peer uploads divided by the total number of video chunks it downloads. In order to understand the impact of NAT on the fairness of a peer-to-peer video

streaming system, we are interested in finding out the average share ratio for public peers and NAT peers, respectively. In an ideal peer-to-peer system, each peer should upload roughly the same amount of data as it downloads, which indicates that the average share ratio for both public peers and NAT peers should be close to 1.

The calculation of share ratio is very straightforward, if the average upload and download amount (in terms of video chunks) can be determined for each group of peers (public and NAT).

For each download request, recall that the probability that it successfully gets video chunk i, $G_p(i)$ and $G_n(i)$ for public peers and NAT peers, respectively, can be calculated using the equations from the previous sections. Therefore, the actual number of downloads in a time slot, or the number of *raw downloads* D'_p and D'_n, are the summation of $G_p(i)$ and $G_n(i)$ multiplied by the number of download requests sent in a slot (D):

$$D'_p = D \sum_{i=1}^{B-1} G_p(i) \tag{3.29}$$

$$D'_n = D \sum_{i=1}^{B-1} G_n(i) \tag{3.30}$$

Note that it is possible that the same chunk is downloaded from different neighbors multiple times, and it counts for multiple raw downloads in this case. It is reasonable not to intentionally avoid requesting the same chunk from neighbors as there is no guarantee that a response will be received for a particular download request.

U'_p and U'_n, the number of *raw uploads* of public peers and NAT peers, can also be determined in a similar approach. The average number of raw uploads of public peers is the sum of total number of downloads by public peers and NAT peers from public peers, divided by N_p, and the average number of raw uploads of NAT peers is

the total number of downloads by public peers from NAT peers divided by N_n:

$$U'_p = \sum_{i=1}^{B-1} \frac{D \cdot G_{pp}(i) \cdot N_p + D \cdot G_n(i) N_n}{N_p} \tag{3.31}$$

$$U'_n = \sum_{i=1}^{B-1} \frac{D \cdot G_{pn}(i) \cdot N_p}{N_n} \tag{3.32}$$

The share ratios for public peers and NAT peers, or SR_p and SR_n, are simply:

$$SR_p = \frac{U'_p}{D'_p} \tag{3.33}$$

$$SR_n = \frac{U'_n}{D'_n} \tag{3.34}$$

3.7 Summary

In this chapter, an analytical model is built for a generic peer-to-peer live video streaming system. The model can be used to quantitatively predict the system performance in terms of video continuity index and share ratio. Although the model is initially built for a generic system with a set of assumptions and simplifications, it can also be extended for other variants of peer-to-peer systems with modifications to some of the modules (discussed in Section 6.1). The model is used to evaluate the system performance and understand the impact of NAT devices, together with simulation results, which validate the analytical model, in Chapter 4.

Chapter 4

Performance Evaluation

In this chapter, system performance evaluation results are presented, including both analytical results and simulation results. The analytical results are obtained from the analytical model built in Chapter 3, and the simulation results come from a software simulator. The evaluation results show the impact on system performance introduced by the existence of NAT devices in the system, in terms of video performance and system fairness. Observations and possible explanations on the impact of NAT devices are included in this chapter, too.

4.1 Software Simulation

Software simulation is a research method that is heavily used in network research. As part of this thesis work, a software simulator is written, so that the performance of a simulated peer-to-peer video streaming system can be measured in order to show the impact of NAT devices. In particular, the simulation is conducted for two network bandwidth scenarios, which look at real-world systems from different angles. Another purpose of having a software simulator is that the simulation results can be compared and cross-checked with the results from the analytical model introduced in

the previous chapter, and the model is validated if the two sets of results are similar, if not identical, to each other.

4.1.1 Software Simulator

The creation of the software simulator used in this thesis work is inspired by some existing simulation works, such as the commonly used BitTorrent simulator created by Bharambe *et al.* [48], and PeerSim which is a popular generic peer-to-peer system simulator [49]. However, existing simulators are not suitable for our research, because they do not include NAT as a key factor and their complicated traffic engine and heavy-weighted communication implementation between simulated peers significantly slow down the simulation runs. While the former problem is relatively easier to solve, the latter requires separating and removing some underlying simulation components, which is a non-trivial and error-prone task. Therefore, a better solution is to write a brand new simulator from scratch, which will be fast and reliable.

The new simulator is written in Java programming language and has no third-party dependency. The simulator uses the same design choices as the generic peer-to-peer live video streaming system described in Chapter 3, e.g., which chunk selection strategy to use. However, the simulation is not just a numerical evaluation of mathematical derivation. Unlike the analytical model which is built for the entire system and is most interested in probability and expectation, the simulator implements each peer as an individual object who maintains its local state (including its own neighbor set, video buffer, etc.) without the knowledge of any global information or the internal state of other peers in the system. A simulated peer is quite realistic, like a light-weighted but fully functional P2P live streaming client software with all essential components, except that it simply counts if a video chunk is available to be played without actually playing it. All the system parameters are tunable by using

a configuration file, and the percentage of NAT peers in the system is the main control variable, as we are most interested in the impact of NAT devices on the system performance. A sample configuration file can be found below.

```
# the configuration parameters are defined as attribute-value pairs
num_peers=1000              # number of peers
num_nat_peers=600           # number of NAT peers
rampup_time=300             # ramp-up time
simulation_time=5400        # simulation time
max_num_neighbors=100       # maximum number of neighbors
max_num_uploads=1           # maximum number of uploads
max_num_downloads=2         # maximum number of downloads
buffer_length=120           # buffer length
tracker_resp_size=50        # number of peers contained in the tracker response
biased_factor=1.0           # biased factor
```

With some code optimization and necessary Java Virtual Machine (JVM) tuning, this simulator is a piece of high-performance software, which is able to simulate large-scale systems consisting of thousands of participating peers. The simulation time is significantly reduced (from over 10 minutes to under 2 minutes for a single simulation run using a typical configuration), which makes it possible to use the simulator to explore various performance improvement schemes (discussed in Chapter 5). In particular, this simulator only implements the application layer and the overlay network of a P2P video streaming system, and all lower layer features (including network impairments such as losses and delays) are ignored. The reason is that in this thesis we are more interested in the logical topology of the system determined by the tracker response and the presence of NAT, rather than the physical topology of the networks. There have been research interests in the literature on constructing

structured P2P systems, e.g., maintaining a mesh- or tree-based structure for video streaming services. However, this is a distinct research topic and not the focus of this thesis work.

Just as the analytical model, video continuity index and share ratio are measured as the major performance metrics. In specific, both performance metrics are measured for public peers and NAT peers as two groups, so that the two groups can be compared with each other and a difference, if any, can be observed. The average continuity index of all peers is also measured, and it is compared with the baseline which is the average continuity index if there is no NAT peer at all. Other metrics, such as neighborhood sizes and response probabilities, are also measured. The purpose is to further validate each of the intermediate steps when building the analytical model.

This software simulator can be made available for research purposes upon requests.

4.1.2 Simulation Scenarios

The simulation scenarios are described as follows. The choices of system parameters are either based on the configurations of real systems, or suggested by other research works.

The total number of peers N is $1,000$, which represents a relatively large-scale system. A certain percentage α of these peers are behind NAT devices and cannot be NAT-traversed successfully. The tracker response contains the information of $K = 50$ peers. There is no limit on the minimum or maximum number of neighbors that a peer maintains.

Analysis in [36] suggests that the performance of BitTorrent-based P2P streaming systems is not sensitive to the size of the peer group for groups larger than 15 to 20 users. Using the above values of N and K guarantees that the group size is always greater than 15, even if up to 75% of the peers are NAT peers. The values of N and

K are proved to have no impact on the system performance by simulation, as long as the system remains a large-scale one (results shown in Section 6.4).

Each peer has a streaming buffer of length $B = 120$, i.e., it holds 120 video chunks (one second each), or 2 minutes of the video. The first 300 seconds of the simulation time is considered to be the ramp-up period and the system performance is not measured during this time. The simulation shows that peers have entered the steady state after the ramp-up period. After that, the real simulation lasts $5,400$ seconds, or 90 minutes, which is the typical length of a full feature movie. The values of these parameters are the typical values that can be found in a real video streaming system.

The network bandwidth setting is a key component of a simulation scenario. In most real-world systems, the average peer uplink bandwidth is merely enough to support the video service, but additional server bandwidth is typically added in order to guarantee a good user experience, which essentially makes such systems a hybrid model of both P2P and client-server paradigms (also known as peer-assisted systems). Since this thesis work is a study on P2P overlay network, we will try to decouple the server factor as much as possible and examine the system in two carefully chosen scenarios in a pure peer-to-peer context. In both scenarios, the extra server bandwidth is removed from the system, leaving only a minimal number of servers to distribute contents. In the first scenario, the typical peer bandwidth setting is kept, which makes the system *bandwidth-constrained*. In the second scenario, the upload bandwidth at each peer is significantly increased so that the system becomes *over-provisioned*. The two simulation scenarios are summarized in Table 4.1.

The upload and download capacity, or U and D, respectively, are configurable parameters to create different scenarios. In the bandwidth-constrained scenario, $U = 1$ and $D = 2$ emulates the link capacities for typical Asymmetric Digital Subscriber

Table 4.1: Summary of Simulation Scenarios

bandwidth-constrained scenario		over-provisioned scenario	
upload capacity	download capacity	upload capacity	download capacity
500 Kbps	1,000 Kbps	1,000 Kbps	1,000 Kbps

Line (ADSL) users (assuming the video playback rate is 500 Kbps, the upload capacity is 500 Kbps and the download capacity is 1,000 Kbps). In this scenario, the average upload bandwidth is the same as the playback rate (1 chunk per second, i.e., 500 Kbps). In the other scenario, $U = D = 2$ emulates the symmetric bandwidth case, and the system is over-provisioned because the upload capacity is twice the video playback rate (1,000 Kbps). In both scenarios, the server upload capacity U_s is set to be 4 times the peer upload capacity, i.e., $U_s = 4U$ (or 2 Mbps in the bandwidth-constrained scenario and 4 Mbps in the over-provisioned scenario).

These two bandwidth scenarios look at real systems from different angles. Authors of [15] studied a real system in which a set (24) of dedicated servers with very high-bandwidth (100 Mbps) are used. Their results have shown that although the video continuity for all peers is good in general (> 0.96), the uploading contribution from peers has a highly unbalanced distribution (30% public peers contribute more than 80% of the total upload bandwidth). Our bandwidth-constrained scenario will be used to find out whether a system with standard peer uplink settings is self-sustainable without much support from video servers. Whether NAT has a strong impact on the fairness of an over-provisioned system is to be verified in our over-provisioned simulation. A more realistic setting capturing the P2P/client-server hybrid model is examined in Section 6.3.

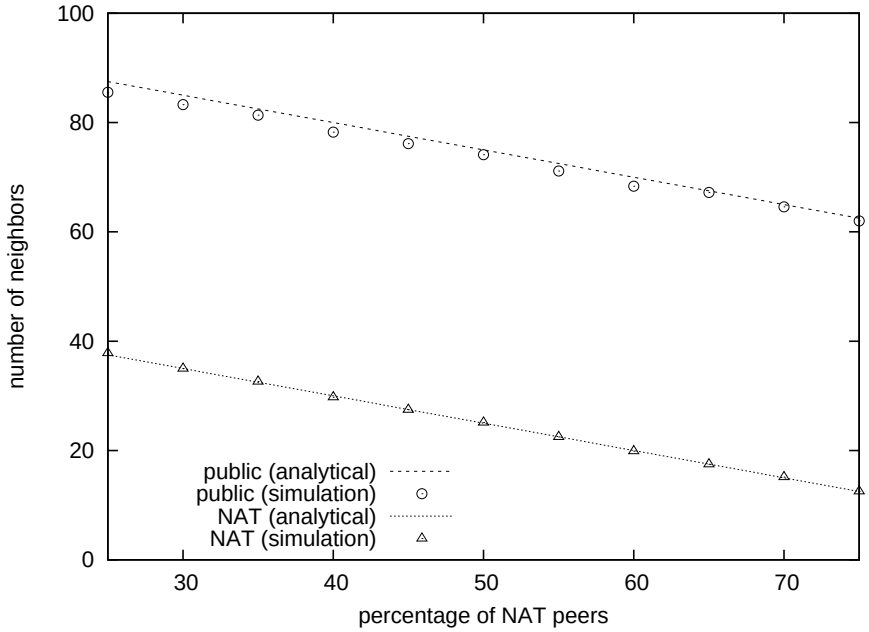

Figure 4.1: Neighborhood Sizes

4.2 Neighborhood Sizes

The analytical model in Chapter 3 reuses part of the previous work on BitTorrent file-sharing systems [44, 45]. Because the system-under-study adopts the tracker-based approach to help new peers join the system, the neighborhood sizes should be the same as in a BitTorrent file-sharing system which also uses the tracker-based approach. In other words, the bootstrapping process should be the same. However, this argument needs to be validated by comparing the reused analytical results with the new simulation results, as shown in Figure 4.1. The analytical results (lines) match with the simulation results (points) closely, which shows that reusing the previous work is reasonable.

Both results show that a public peer always has a larger neighborhood size than a NAT peer. This is because a public peer can not only initiate connections to other peers, but also receive incoming connection requests from other peers if it is included in the tracker response to a newly joined peer, while a NAT peer can only be the one

who initiates connections. For this reason, it is possible that the neighborhood size of a public peer is greater than the number of peers returned by the tracker K. It is also shown in the figure that as the percentage of NAT peers α grows, the neighborhood sizes for both public peer and NAT peer decrease. The reason is that the existence of more NAT devices further limits the connectivity in the system.

The difference in neighborhood size between public peers and NAT peers has been observed in preliminary experiment results using real-world P2P live streaming software. More systematic and conclusive results are to be presented in a technical report following this thesis work.

4.3 Response Probabilities

It is very likely that public peers and NAT peers have different probabilities of responding to a download request, because of the difference in their neighborhood sizes and the level of contentions.

The analytical model built in Chapter 3 is able to predict the response probabilities. The analytical results are plotted together with the simulation results in Figure 4.2 and Figure 4.3, each representing the results on one of the two scenarios mentioned in the previous section.

Both figures show that NAT peers have a higher response probability than public peers, because they have fewer neighbors and it is less likely to see contentions for the upload capacity of a NAT peer. In the over-provisioned scenario, where a peer can support 2 uploads simultaneously, NAT peers can almost respond to all download requests received, indicated by a response probability that is close to 1 (the baseline in the figures). When there are more NAT peers in the system, the gap between public peers and NAT peers in response probabilities is even bigger, because of the even

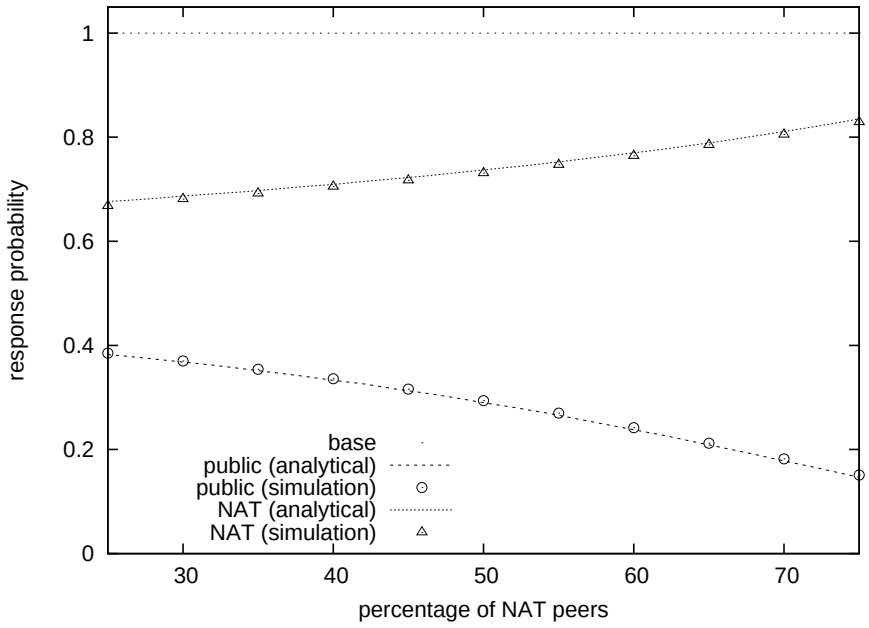

Figure 4.2: Response Probability (bandwidth-constrained)

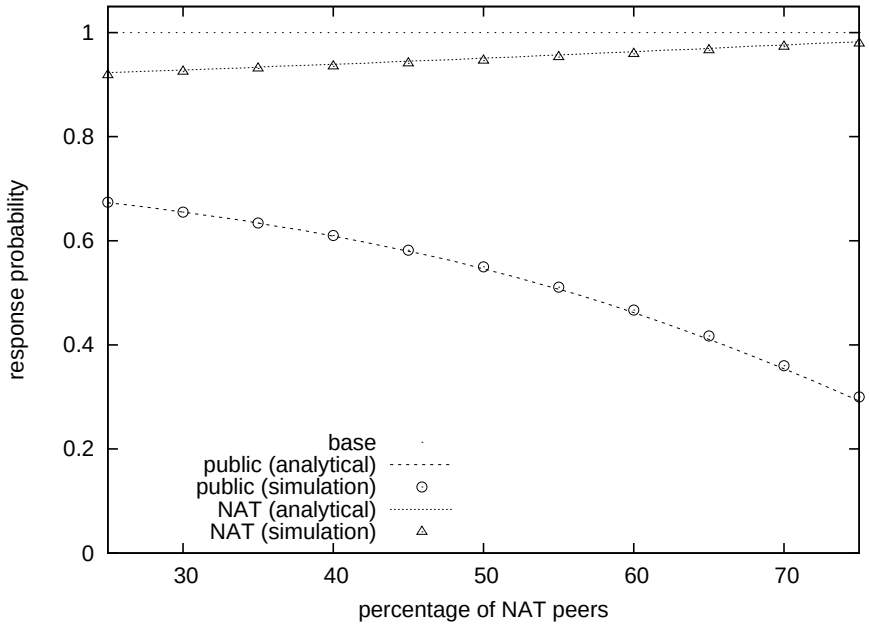

Figure 4.3: Response Probability (over-provisioned)

more intensified contentions for the upload capacity of public peers. The implication of such results on response probabilities is that reducing the contention for the upload bandwidth of public peers, as well as redirecting more download requests to NAT peers to make use of their high response probability, is among the major tasks if the system performance is to be improved.

In both figures, the analytical results and the simulation results (points and lines) match with each other very well, which means that the analytical model can precisely predict the response probabilities.

4.4 Video Continuity

During the simulation, the continuity index of each peer is measured, and then the average continuity index is calculated for all peers, public peers only, and NAT peers only. The simulation results are plotted together with the analytical results obtained from the analytical model, for both scenarios described in the previous section, in Figure 4.4 and Figure 4.5. The horizontal straight line in the figures is the baseline which is the average continuity index when there is no NAT peer in the system at all.

In general, the existence of NAT devices significantly degrades the performance of a peer-to-peer video streaming system. As the percentage of NAT peers in the system α increases, the overall continuity drops, especially for NAT peers. In the bandwidth-constrained scenario (see Figure 4.4), while the continuity of public peers is reasonable, the continuity of NAT peers is almost unacceptable (lower than 0.7). Therefore, having servers in an otherwise bandwidth-constrained peer-to-peer system is necessary to provide satisfactory video services. Results in Section 6.3 show that with enough help from extra servers, the video continuity for both public and NAT peers can be brought back to a satisfactory level and the difference between the two

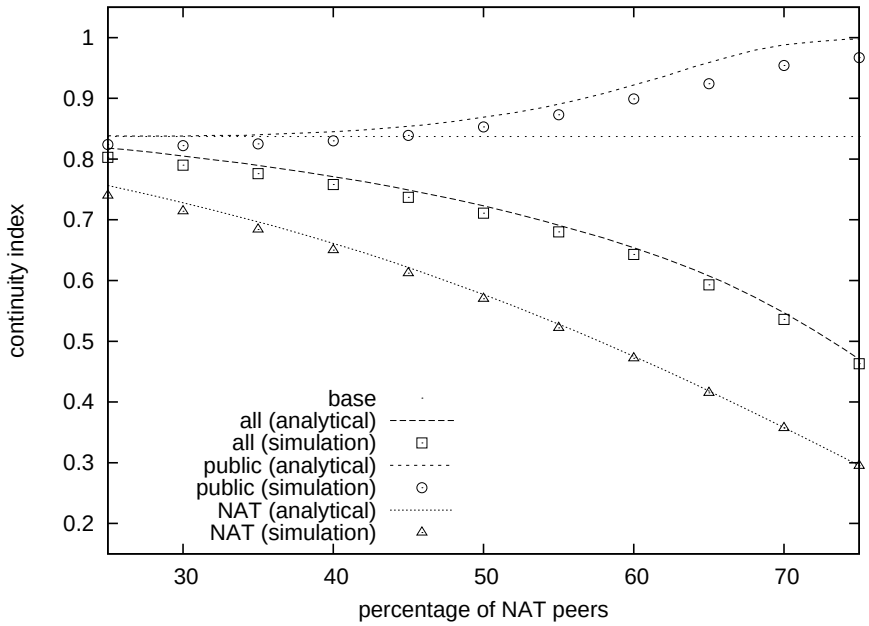

Figure 4.4: Continuity Index (bandwidth-constrained)

groups will be almost completely gone. Contribution from servers is also the reason why the performance difference between public peers and NAT peers is not observed by measurement work on real systems. It can be inferred from our results that most of the added server bandwidth is used for helping NAT peers for better performance. Also, a direct cause to the performance difference is the difference in neighborhood sizes as shown in a previous section. Letting NAT peers have more neighbors can help improving the video performance for NAT peers, but it is not the best way for performance improvement, which will be discussed in more detail in later chapters.

An interesting observation on simulation results from the bandwidth-constrained scenario is that public peers benefit from the existence of NAT peers by experiencing a better continuity. The reason is that NAT peers are contributing their upload capacity to public peers exclusively, and this behavior is similar to the observation in the previous work [44, 45] on the impact of NAT on peer-to-peer file-sharing systems.

If the system is over-provisioned by peer upload bandwidth (see Figure 4.5), the

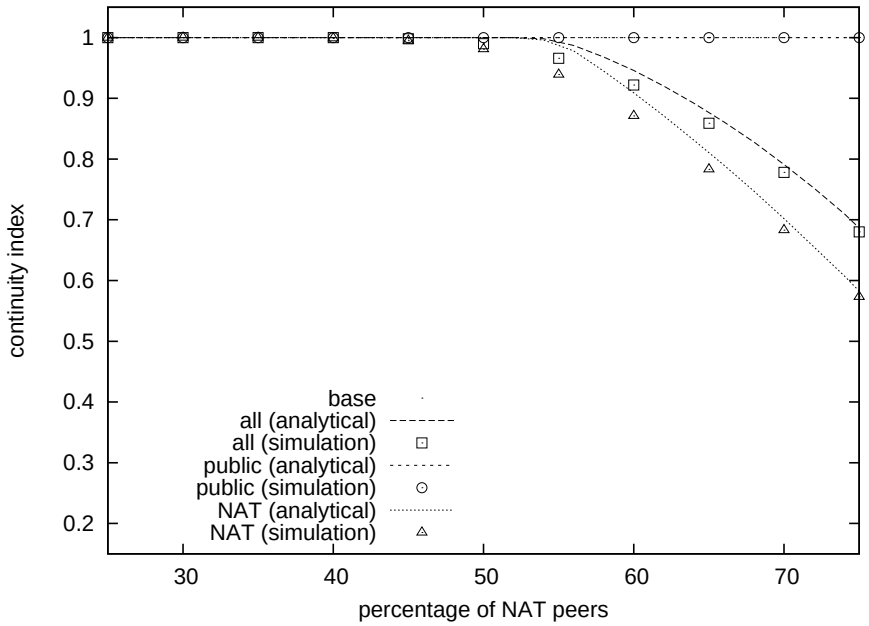

Figure 4.5: Continuity Index (over-provisioned)

overall performance of all peers, including that of NAT peers, is quite good (continuity index ≈ 1) even when there are a considerable number of NAT peers in the system. However, when the percentage of NAT peers exceeds a certain threshold, around 50% in this particular scenario, the system is no longer self-sustaining and the average continuity starts to drop again. Note that in the definition of NAT peers there is an extra condition of unsuccessful NAT traversal, and 50% is actually a fairly high percentage in real systems.

50% is a critical point for this specific scenario because the average peer upload bandwidth is twice the streaming rate in our over-provisioned setting. At this point (50%), there are equal number of public peers and NAT peers, and the total upload bandwidth of public peers is just enough to support both groups (public and NAT). Going beyond this point, NAT peers as a group will not get enough upload resources and their performance will begin to degrade. In [41], 50% is also considered as a threshold above which it is impossible to avoid the fairness issue in peer-to-peer

systems (the firewalled peers in their work are equivalent to NAT peers in this thesis). In addition, 50% is a critical point in the performance improvement scheme proposed in this thesis, the reason of which will be given in Chapter 5.

The fact that there is only small gap between analytical results and simulation results validates the analytical model. The discrepancy should be caused by the randomness in the simulator, which can be verified by examining the logs generated during the simulation. Data entries for 10 peers in the log file from a simulation run are shown below. While the continuity index of those peers enjoying good video performance is upper-bounded by 1, an unlucky peer (peer #700 for instance) may experience very poor video performance due to simulation randomness and the average continuity index of all peers is lowered because of that. This does not have any major statistical impact, but its impact is big enough to cause the small discrepancy between the two sets of data. The fact that the simulation results do not perfectly match with the analytical results shows that the simulation is not just a numerical evaluation of mathematical derivation.

ID	Type	Continuity	ID	Type	Continuity
...	...	...	700	PUB	**0.867**
695	PUB	0.937	701	PUB	0.925
696	PUB	0.986	702	PUB	0.930
697	PUB	0.948	703	PUB	0.948
698	PUB	0.970	704	PUB	0.939
699	PUB	0.981	...	...	...

4.5 System Fairness

A peer's share ratio is defined as its total upload amount divided by its total download amount. In a fair system, the share ratio of all peers should be around 1, which means

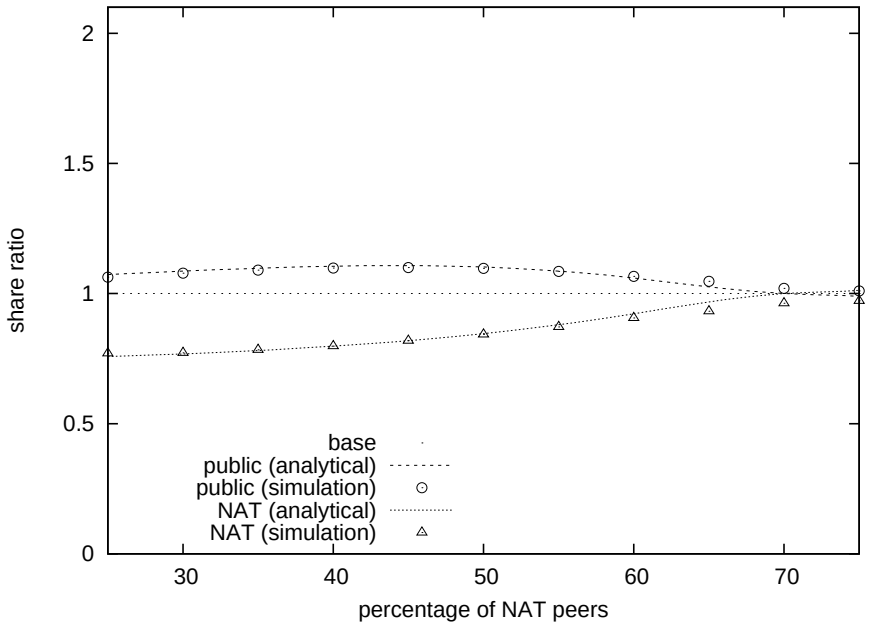

Figure 4.6: Share Ratio (bandwidth-constrained)

each peer uploads approximately the same amount of data as it downloads.

During the simulation, the share ratio of each peer is also measured as a fairness metric, in addition to the continuity index mentioned in the previous section. Again, the average for public peers and NAT peers are calculated separately for comparison purposes. Simulation results for the average share ratio of both public and NAT peers, for both scenarios, are plotted in Figure 4.6 and Figure 4.7, respectively. The simulation results fit the analytical results quite well again, which shows that the analytical model can be used to predict the share ratio, too.

In these figures, it can be observed that public peers have an average share ratio greater than 1, while the average share ratio of NAT peers is less than 1. This indicates that public peers upload more than they download while NAT peers download more than they upload, which is on the contrary of the observation in BitTorrent file-sharing systems [44, 45]. This is due to the streaming constraint that keeps public peers in the system to serve NAT peers, rather than to finish downloading the file

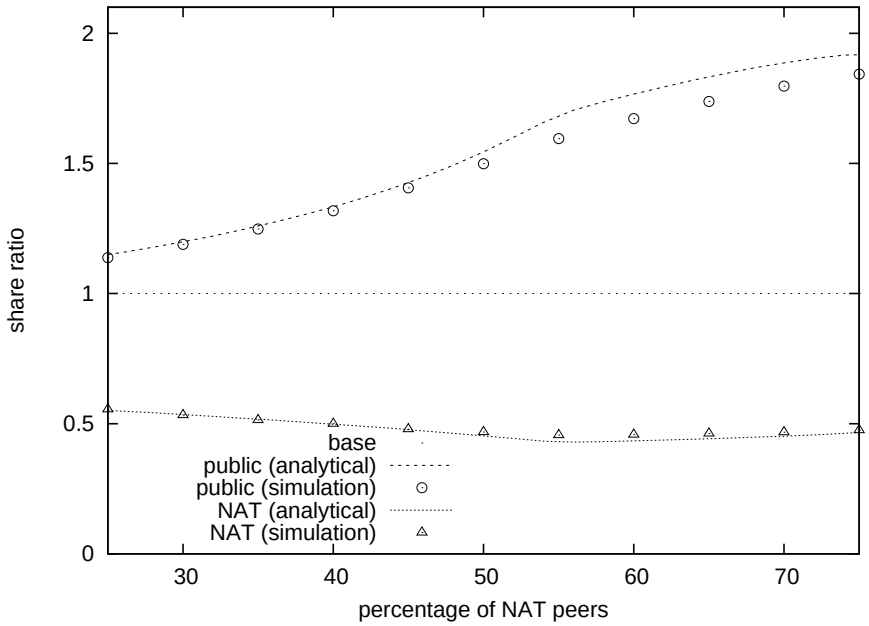

Figure 4.7: Share Ratio (over-provisioned)

and leave. Furthermore, the lower continuity of NAT peers indicates that NAT peers have a lower probability of having video chunks in the buffer. Because of its poor video content availability, it is less likely that a NAT peer is able to contribute to public peers by having the chunks that are missing from its public neighbors' buffer, resulting in a lower share ratio for NAT peers. This also highlights the difference between file-sharing systems and video streaming systems.

In the bandwidth-constrained scenario, the difference in the share ratio between public peers and NAT peers is not significant. However, in the over-provisioned scenario, the share ratio of public peers is much greater than that of NAT peers (almost by 3 times for a larger α). Recall that the continuity of public peers in this scenario is always good when $\alpha < 50\%$, public peers are actually uploading to other peers way more than they download, and the fairness becomes the major issue.

Results from the over-provisioned simulation are supported by measurement studies in [15]. Although the way in which the system is over-provisioned is different, i.e.,

increased peer uplink capacity versus extensive use of high-capacity servers, both systems are over-provisioned from a viewpoint at the system level. Note that the percentage of NAT users in [15] is around 70%, but the NAT peers in this thesis have an extra condition of unsuccessful NAT-traversal. When NAT traversal works for more than 25% of the time (which is reasonable according to [8, 9, 10]), the results in [15] falls on the left half of Figure 4.5 and Figure 4.7, which have also shown a close-to-one continuity index for all peers but a significant fairness issue.

4.6 Summary

Performance evaluation results from both the analytical model and software simulation are presented in this chapter. The analytical model is shown to be capable of precisely predicting the system performance, validated by the simulation results. The system is evaluated for two specific scenarios. Results show that the existence of NAT devices degrades the overall video continuity and breaks the fairness across the peer-to-peer system. In particular, NAT peers experience a low continuity index while a high share ratio is expected for public peers. In a system where the upload capacity is a constraint, the degradation in continuity is severe; while in an over-provisioned system, the fairness issue becomes a major concern.

Chapter 5

Performance Improvement

Chapter 4 illustrates that NAT has negative impacts on peer-to-peer video streaming systems, by degrading system performance and breaking system fairness. To alleviate such negative impacts, a simple but effective mechanism is proposed in this chapter, by introducing minimum changes to the existing system modules. The mechanism is based on biased peer selection, which is explained in detail in this chapter. This chapter also shows how to find the optimal bias factor, which is a critical parameter for the biased peer selection scheme, as well as the effectiveness of the proposed performance improvement mechanism.

5.1 Biased Peer Selection

In essence, NAT degrades the system performance in two ways:

(a) The high contention for the upload capacity of public peers causes NAT peers to experience a low response rate from their (public) neighbors; and

(b) The upload capacity of NAT peers is not fully utilized because of their low availability of video chunks.

In this thesis, a *biased peer selection* mechanism is proposed, in attempt to coun-

teract both factors and improve the system performance.

The difference in neighborhood sizes between public peers and NAT peers is a major factor in NAT's negative impact on the system, and the most straightforward solution is to increase the neighborhood size for NAT peers. For example, if the tracker returns a larger list of active peers to NAT peers than to public peers, NAT peers will have more neighbors and have a better chance to enjoy better performance. However, a straightforward solution is not necessarily the best solution. Compared with the biased peer selection scheme presented in this chapter, although this approach is simpler in terms of implementation, it introduces more network overhead and requires more computational and storage resources on the client software, and does not improve system performance as much when the percentage of NAT peers is high. A comparison between the two schemes is available in Section 6.5.

In previous chapters, the system-under-study uses the random peer selection strategy, where a peer just uniformly selects some of its neighbors at random and sends its download requests to each of them. Random peer selection makes no distinction between public neighbors and NAT neighbors.

Recall that the probability that a public peer selects a public neighbor to send a download request ρ_{pp}, the probability that a public peer selects a NAT neighbor ρ_{pn}, and the probability that a NAT peer selects a (public) neighbor ρ_n can be calculated as follows:

$$\rho_{pp} = \rho_{pn} = \frac{1}{L_p} \tag{5.1}$$

$$\rho_n = \frac{1}{L_n} \tag{5.2}$$

The peer selection strategy can be modified so that the system performance is improved. In a biased peer selection strategy, different weights are assigned to public neighbors and NAT neighbors. Note that the rule only applies for public peers, as

NAT peers only have public neighbors. Define the *bias factor*, denoted by β, as follows:

$$\rho_{pp} = \beta \cdot \rho_{pn} \tag{5.3}$$

If $0 \leq \beta < 1$, NAT peers are given higher priorities when a public peer selects a neighbor to send its download requests (thus *biased*). ρ_{pp} and ρ_{pn} can be solved by Equation (5.3) along with the equation below:

$$\rho_{pp} L_{pp} + \rho_{pn} L_{pn} = 1 \tag{5.4}$$

As no modifications are introduced to NAT peers, $\rho_n = 1/L_n$ still holds.

Random peer selection is a special case of biased selection, where $\beta = 1$.

The intuitions of this mechanism are:

(a) By directing some of the download requests from public peers to their NAT neighbors, the contention for the upload capacity of public peers is alleviated, so that NAT peers have better chances to get responses to their download requests; and

(b) NAT peers receive more download requests, so that their upload capacity can be better utilized.

The biased peer selection mechanism is viable and it only requires minimum modifications to the existing client software and the tracker. It is possible for the client software to determine whether the host PC is on a private network by examining the local IP address (private IP addresses are in the range of 10.x.x.x, 172.16.x.x through 172.31.x.x, or 192.168.x.x). The measurement work in [15] uses such an approach to report the percentage of private users in their system. Alternatively, STUN [5] can be used to discover the presence of a NAT device. This information is sent to the tracker by the client software in the join request when a new peer joins the system. The tracker collects the global information and calculates the percentage of

NAT peers alive in the system (α), and returns it to the joining peer along with the partial peer list. The new peer computes the ρ values based on α, and assign weights to its neighbors accordingly after its neighborhood is built. To deal with network dynamics, the tracker can periodically push the α value to each peer, possibly in the keep-alive messages, and the weights are then re-calculated and updated locally by the client software. The extra network overhead introduced by this mechanism is negligible, because it re-uses existing messages and only adds one parameter which has a constant size.

5.2 Optimal Bias Factor

The choice of the bias factor β is crucial for the biased peer selection mechanism. A bias factor is considered to be *optimal* if both of the following conditions are met at the same time:

(a) The overall average continuity index is maximized, and

(b) The average share ratios for public peers and NAT peers are both close to 1.

The analytical model built in Chapter 3 can be used to evaluate a bias factor. The only change needed is to substitute the equations computing the ρ values, using the new values as functions of β (see the previous section).

In order to illustrate that there exists an optimal bias factor, presented below are the results from the analytical model and software simulation for a specific system where 30% of the peers are NAT peers, i.e., $\alpha = 30\%$. Just like previous chapters, the results for both the bandwidth-constrained scenario and the over-provisioned scenario (see Table 4.1) are captured.

For the bandwidth-constrained scenario, Figure 5.1 and Figure 5.2 show the continuity index and share ratio metrics when different values of the bias factor β are

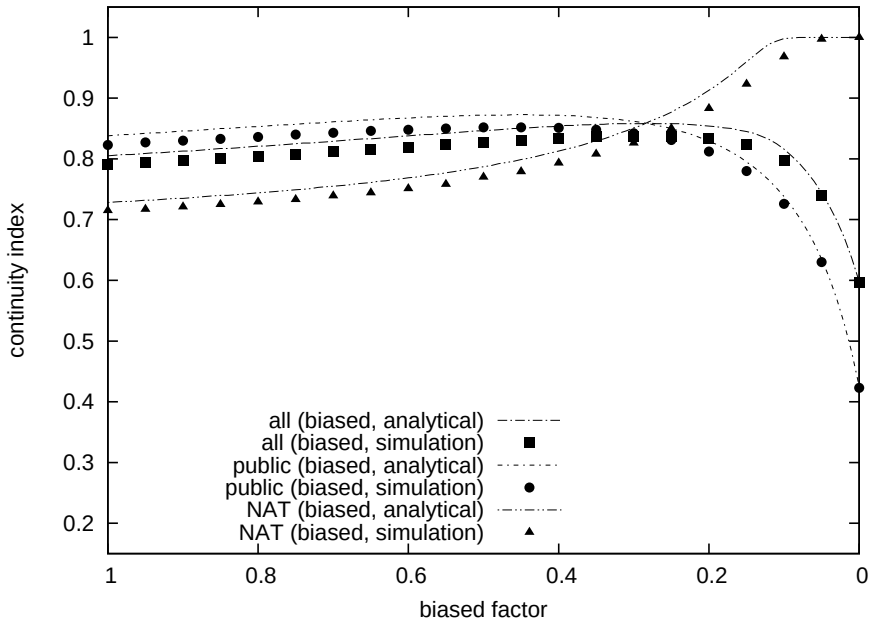

Figure 5.1: Biased Factor (continuity index, bandwidth-constrained)

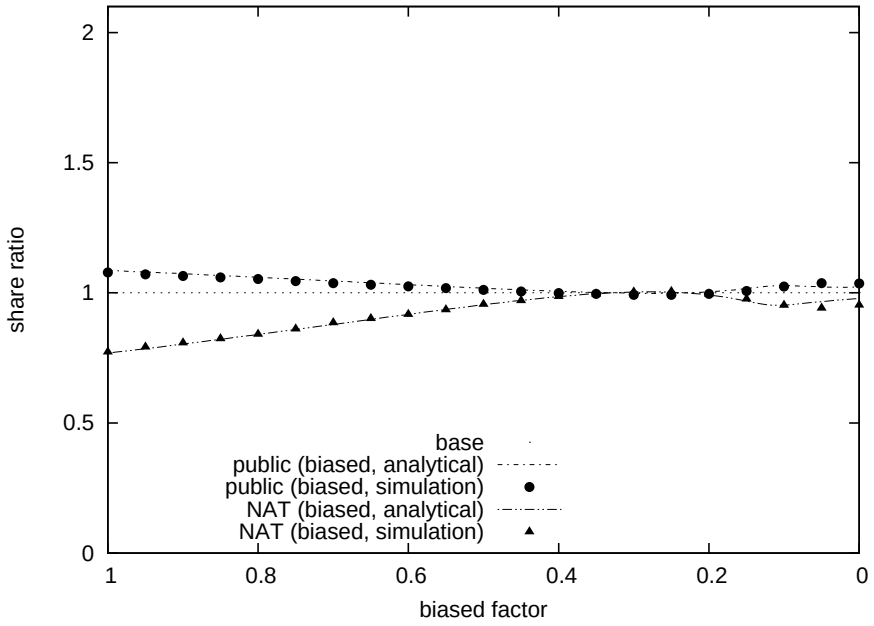

Figure 5.2: Biased Factor (share ratio, bandwidth-constrained)

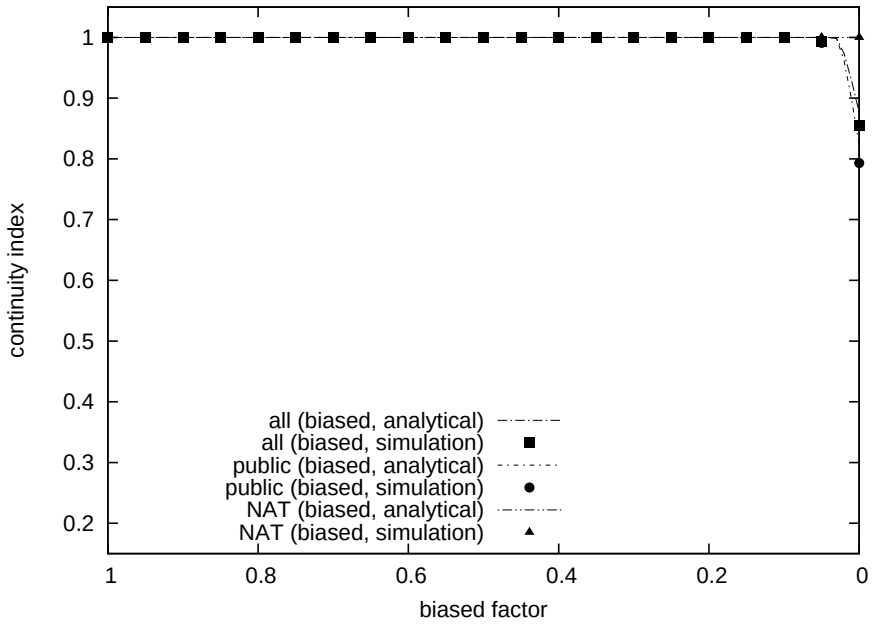

Figure 5.3: Biased Factor (continuity index, over-provisioned)

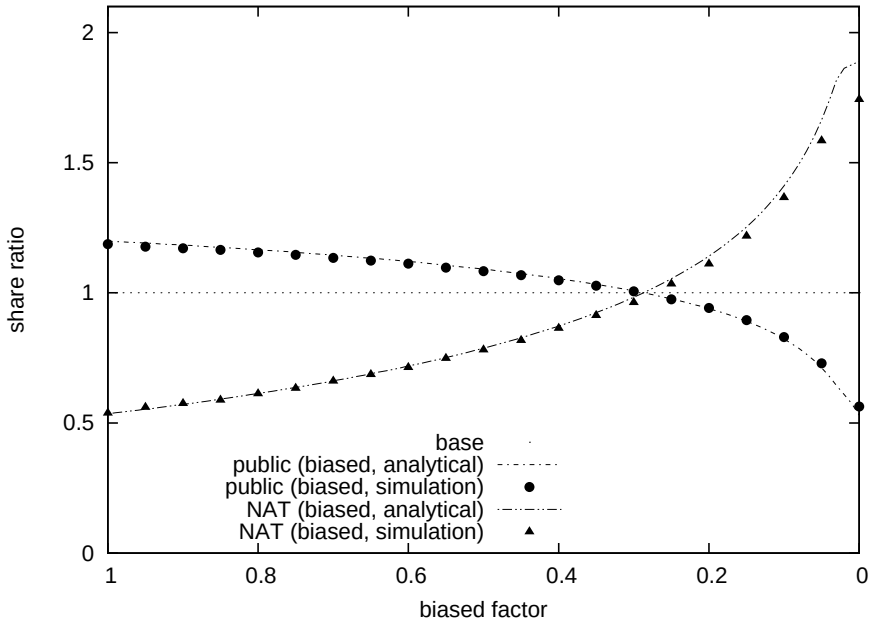

Figure 5.4: Biased Factor (share ratio, over-provisioned)

used. β is always a value between 0 (inclusive) and 1 (exclusive), which means the peer selection is biased towards NAT peers. Both analytical and simulation results suggest that the overall continuity index is maximized if the bias factor is 0.28 (see Figure 5.1). At this point, the average continuity index of public peers is approximately the same as that of NAT peers, too. Figure 5.2 shows that 0.28 is also in the range where the share ratios for both public peers and NAT peers are both very close to 1, indicating that the system is a fair one. Therefore, 0.28 is a close-to-optimal bias factor for this scenario if 30% of the peers are NAT peers for this scenario.

Similarly, Figure 5.3 and Figure 5.4 show the continuity index and share ratio results for the over-provisioned scenario. It is difficult to determine the bias factor to achieve the best video continuity, as the continuity index is always very close to 1. However, if $\beta = 0.28$, the average share ratio is close to 1 for both public peers and NAT peers, and clearly this is the bias factor that achieves the best fairness.

It appears that for a given percentage of NAT peers in the system α, the optimal bias factor β is the same for both scenarios. This is because no matter in what bandwidth scenario, the biased peer selection scheme always re-distribute the upload bandwidth between public peers and NAT peers in the same way, given the same β value. Since our scheme redirects download requests at the system level, independent of peer bandwidth, the optimal β will be the same for all bandwidth scenarios, including the bandwidth-constrained and over-provisioned scenarios shown above.

In all four figures, the analytical curves always closely match with the simulation data, meaning that it is reliable to use the analytical model to evaluate β. As such, for α values between 25% and 75%, the analytical model is used to calculate the continuity index and share ratio for all β values between 0 and 1. In this way, the bias factors leading to the best performance given each α value are determined. The optimal β values based on empirical data for each α are listed in the second row of

Table 5.1: Optimal Bias Factors

α	25%	30%	35%	40%	45%	50%
optimal β (empirical)	0.33	0.28	0.24	0.16	0.09	0.00
optimal β	0.3333	0.2857	0.2307	0.1666	0.0909	0
α	55%	60%	65%	70%	75%	
optimal β (empirical)	0.00	0.00	0.00	0.00	0.00	
optimal β	0	0	0	0	0	

Table 5.1.

Empirical data suggests that there is a different optimal β for each α value. However, a mathematical expression is needed to calculate the optimal β value for any given α. The ultimate goal of biased peer selection is to redirect download requests, so that each peer receives an equal number of requests, which further leads to equal response probabilities in a homogeneous setting. Since the total number of download requests per time slot is $(D \cdot N)$, the target is that each peer receives $(D \cdot N)/N = D$ requests on average. In other words, $Q_p = Q_n = D$. ρ_{pp} and ρ_{pn} can be solved by manipulating Equations (3.9) and (3.10).

$$Q_p = D \cdot \frac{(\rho_{pp} L_{pp}) N_p + N_n}{N_p} = D \tag{5.5}$$

$$Q_n = D \cdot \frac{(\rho_{pn} L_{pn}) N_p}{N_n} = D \tag{5.6}$$

$$\rho_{pp} = \frac{N_p - N_n}{L_{pp} N_p} = \frac{1 - 2\alpha}{2(1-\alpha)^2 K} \tag{5.7}$$

$$\rho_{pn} = \frac{N_n}{L_{pn} N_p} = \frac{1}{(1-\alpha) K} \tag{5.8}$$

Then β can be calculated by definition:

$$\beta = \frac{\rho_{pp}}{\rho_{pn}} = \frac{1 - 2\alpha}{2(1-\alpha)} \quad (\alpha \leq 0.5) \tag{5.9}$$

The optimal β values given by this expression (the third row of Table 5.1) are very close to those from empirical data. Note that when α is greater than 50%, this expression gives a negative value. Since the value of β cannot be negative by definition, the optimal β is always 0 when $\alpha > 50\%$. $\beta = 0$ means that public peers only send their download requests to NAT neighbors, but even so, the two conditions on the optimal biased factor are not satisfied when over 50% of the peers are NAT peers. In that sense, the system is still not optimal, which means that the biased peer selection mechanism alone does not always completely counteract the negative impact of NAT when the percentage of NAT peers is very high (more results and discussions on why 50% is a critical point in the next section).

5.3 Effectiveness of the Mechanism

In the previous section, a set of optimal bias factors are determined, using which values the biased peer selection mechanism performs the best. However, how much the mechanism can help improve the system performance is still uncertain. This section contains the analytical and simulation data of the improved video continuity and share ratio, in comparison with the known data (see Chapter 4) for the original system, so as to show the effectiveness of biased peer selection proposed in this thesis.

In Figure 5.5, which shows the improved continuity index for the bandwidth-constrained scenario, the new results with biased peer selection enabled are marked by filled points, while the previous results are marked by unfilled points. The figure shows that the continuity index for both public peers and NAT peers has improved, hence an improved overall continuity as well (not shown here to avoid too many data points being too close to each other, but the results are available in the raw data). If no more than a half of the total peers are NAT peers, the continuity for all peers is

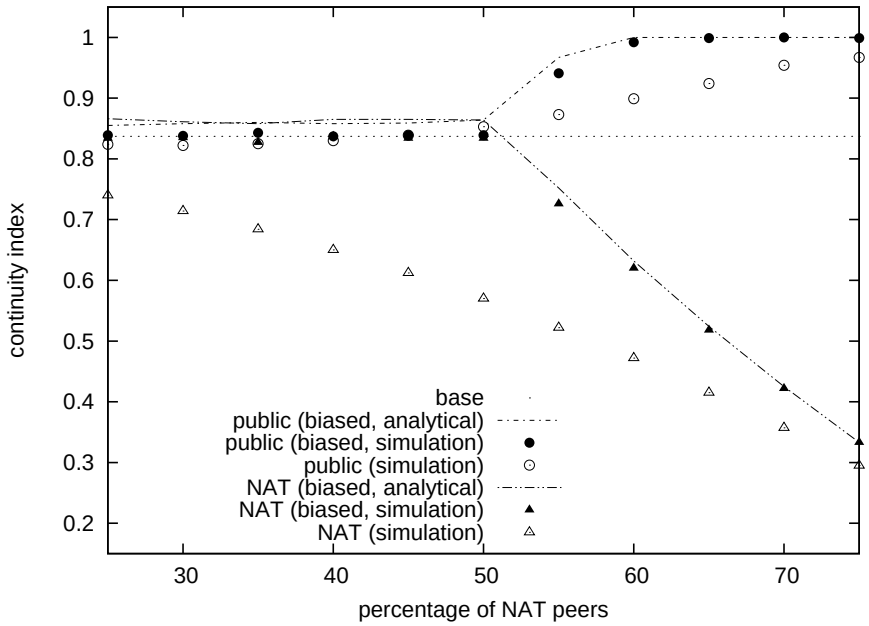

Figure 5.5: Improved Continuity Index (bandwidth-constrained)

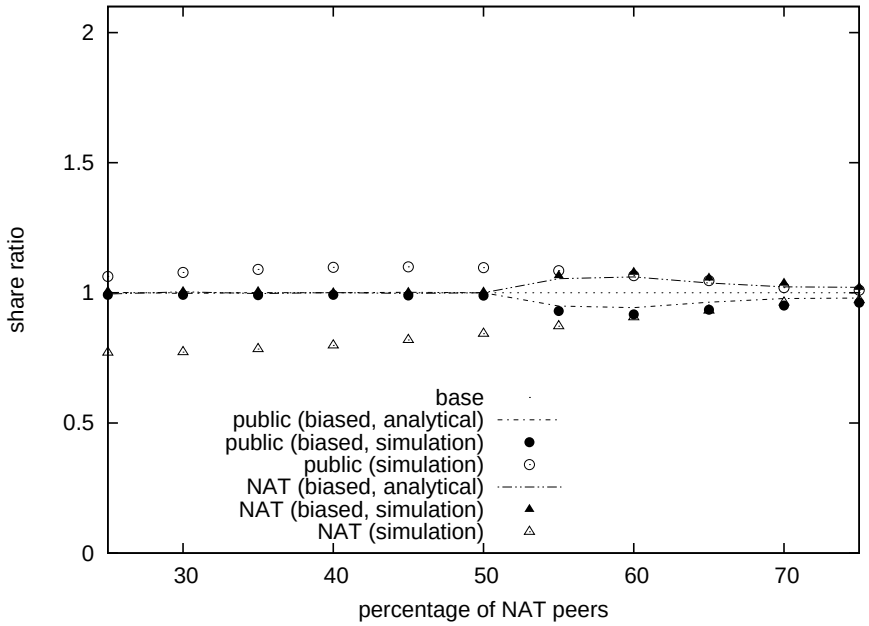

Figure 5.6: Improved Share Ratio (bandwidth-constrained)

approximately the same as the baseline, which is the average continuity index when there is no NAT peers at all. If NAT peers are more than a half of total peers, the continuity of NAT peers begins to drop, and it is much worse than the continuity of public peers. But still, the performance is better than that without biased peer selection.

The improved share ratio for the same scenario is shown in Figure 5.6. With less than 50% NAT peers, both public peers and NAT peers have a share ratio very close to 1, indicating good system fairness. The improvement in share ratio is quite limited when there are more than 50% NAT peers, but as discussed before, share ratio is a less critical issue in a bandwidth constrained scenario.

Figure 5.7 and Figure 5.8 show the performance improvement in a similar way for the over-provisioned scenario. The continuity index is a less critical issue in this scenario, but an improvement in the continuity of NAT peers still can be seen. There is a huge gap between the share ratio for public peers and NAT peers without biased peer selection. After enabling the biased peer selection mechanism, the gap is completely gone for α values that are less than 50%. The gap is narrowed even for larger α values.

$\alpha = 50\%$ is a critical point in the effectiveness of the biased peer selection scheme. The reason is that when $\alpha = 50\%$ and $\beta = 0$, essentially all public peers are contributing their upload bandwidth to NAT peers, while NAT peers always upload to public peers exclusively. Since the number of public peers is equal to that of NAT peers in this case, each peer gets roughly the same upload bandwidth from other peers, and the whole system is balanced and experiences good overall performance. However, when α goes beyond 50%, even though public peers are still contributing all their upload capacity to NAT peers ($\beta = 0$), there are more NAT peers than public peers and they cannot all get enough bandwidth to consume. Therefore, although the

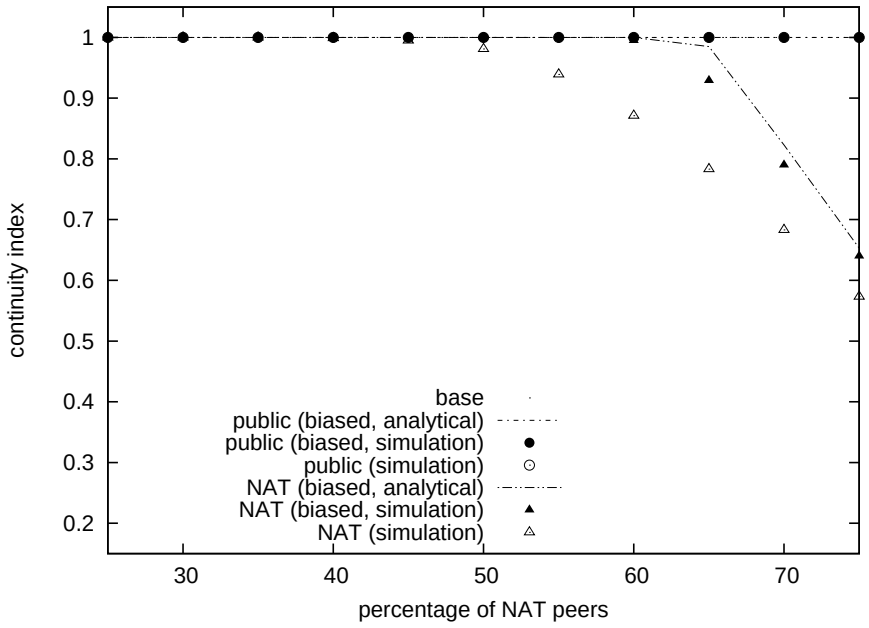

Figure 5.7: Improved Continuity Index (over-provisioned)

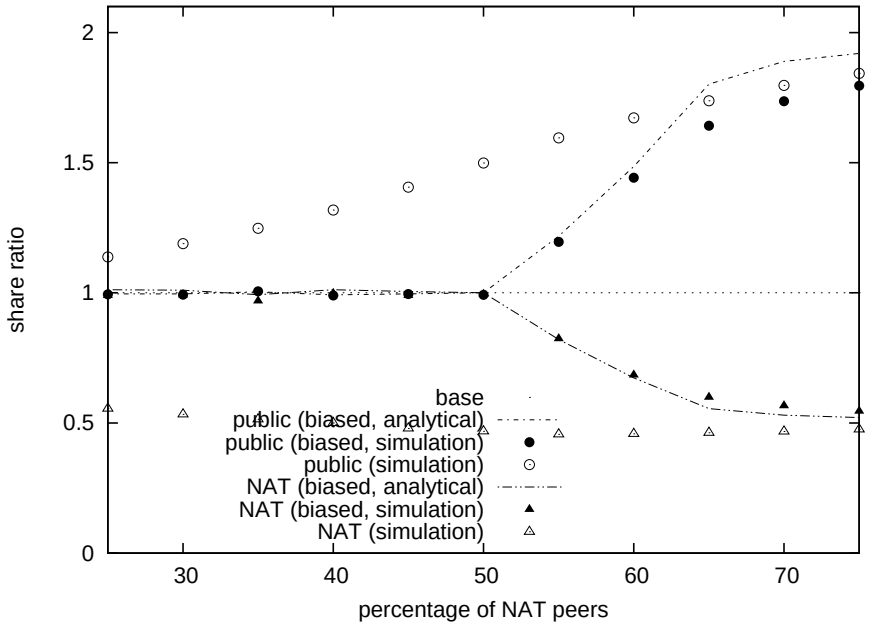

Figure 5.8: Improved Share Ratio (over-provisioned)

biased peer selection scheme alone can still improve the system performance, optimal performance cannot be achieved. This characteristic is a result of the way our scheme works which is specific to biased peer selection, and the observations on $\alpha = 50\%$ do not apply to other schemes or other systems.

Combining biased peer selection with some other schemes can help further improve the system performance and fairness, even achieving optimal performance. For example, the biased peer selection can also be introduced to the video server(s), so that even more upload bandwidth is allocated to NAT peers from the server side. Another possibility is to introduce some rate limit mechanism on the download speed of some (NAT) peers.

5.4 Summary

In this chapter, a biased peer selection mechanism is proposed to improve the system performance (which has some similarity with the *biased opportunistic unchoke* strategy proposed for BitTorrent file-sharing systems in the previous research work [44, 45]). Only minimum modifications to the existing system and client software are involved in this mechanism. A set of optimal bias factors are determined for the scheme, using which both maximized continuity index and narrowed gap between public peers and NAT peers in their share ratio can be achieved at the same time. Analytical and simulation results justify that the mechanism is effective in improving system performance, especially when the number of NAT peers is less than half the total number of peers. The system performance can be further improved if biased peer selection is used in combination with some other schemes.

Chapter 6

Further Discussions

Some specific topics are discussed in this chapter. These topics either support some claims made in earlier chapters, or explore alternatives to the assumptions or simplifications made for the analytical model and the simulation. Such topics include illustrations on how the analytical model can be conveniently extended using two examples, and a study on gossip-based membership protocol, as well as discussions how using extra servers can improve system performance and whether the performance improvement scheme in the previous chapter can effectively reduce the server bandwidth usage. Also available in this chapter are discussions on whether the size of the system and the amount of information returned in a tracker response have any effect on simulation results, and if a biased tracker scheme is a good alternative as a performance improvement mechanism.

6.1 Extending the Analytical Model

Although the analytical model in Chapter 3 is built for a generic peer-to-peer video streaming system, it is possible to extend the model to capture the performance of other system variants. The flexibility is resulted from the fact that the modules of

the model, such as neighborhood size, peer selection, chunk selection, etc., are loosely coupled with each other. In order to illustrate that the model is a highly extensible one, two examples are included in this section, one for systems using a different chunk selection scheme and the other for heterogeneous network conditions.

6.1.1 Greedy Chunk Selection

In Chapter 3, the system-under-study adopts a rarest-first strategy for video chunk selection. Rarest-first means when a peer requests video chunks from its neighbors, it always asks for those missing chunks with the lowest availability in the system, which are the ones that are further away from the playback point in the video buffer. The rarest-first scheme is intended to increase the diversity of data availability in the system, hence improving the system performance (particularly large-scale systems). Rarest-first is used as the file piece selection strategy in the BitTorrent file-sharing application.

Greedy, or sequential, chunk selection strategy is an alternative to the rarest-first strategy, the idea of which is quite straightforward. In a system using greedy chunk selection, a peer always tries to download the missing chunks that are closest to the playback point, so as to increase the chances of meeting the playback deadline. The greedy strategy is quite intuitive, and users can experience a shorter startup delay before they can start viewing the program, when compared with rarest-first. In addition, the greedy scheme performs better than rarest-first when the peers have a limited buffer space on average [37]. A combination of rarest-first and greedy chunk selection strategy is also possible.

The analytical model can be extended for systems using the greedy chunk selection strategy with only small modifications. In the existing model for the video buffer, the probabilities that a peer selects a particular chunk i to download from its

neighbor $S_{pp}(i)$, $S_{pn}(i)$, and $S_{np}(i)$ are calculated for the rarest-first strategy, using Equations (3.17), (3.18) and (3.19). For the greedy strategy, the conditions that a peer requests a particular chunk i to download from its neighbor are different:

(a) Chunk i is missing from the buffer of the requesting peer but is available in the requested peer's buffer; and

(b) For chunks $i+1$ through $B-1$ (the last slot in the buffer before the playback deadline), either the peer already has it in the buffer, or neither the peer itself nor the requested peer has it in the buffer.

These equations can be modified accordingly as follows, in order to capture the greedy chunk selection strategy.

$$S_{pp}(i) = (1 - H_p(i))H_p(i) \cdot \prod_{j=i+1}^{B-1} (H_p(j) + (1 - H_p(j))(1 - H_p(j))) \tag{6.1}$$

$$S_{pn}(i) = (1 - H_p(i))H_n(i) \cdot \prod_{j=i+1}^{B-1} (H_p(j) + (1 - H_p(j))(1 - H_n(j))) \tag{6.2}$$

$$S_{np}(i) = (1 - H_n(i))H_p(i) \cdot \prod_{j=i+1}^{B-1} (H_n(j) + (1 - H_n(j))(1 - H_p(j))) \tag{6.3}$$

Because of the low dependency between different modules of the analytical model, such changes do not break other modules and they are the only modifications needed to extend the model for the greedy strategy. The performance of a system using greedy strategy can be easily evaluated using the extended analytical model.

Since the greedy chunk selection strategy is merely used as an example illustrating that the analytical can be easily extended, only preliminary results of this modified model are presented here. The continuity index results of the bandwidth-constrained scenario for greedy chunk selection using the modified model are shown in Figure 6.1.

In this figure, the points are the analytical results for the original model with

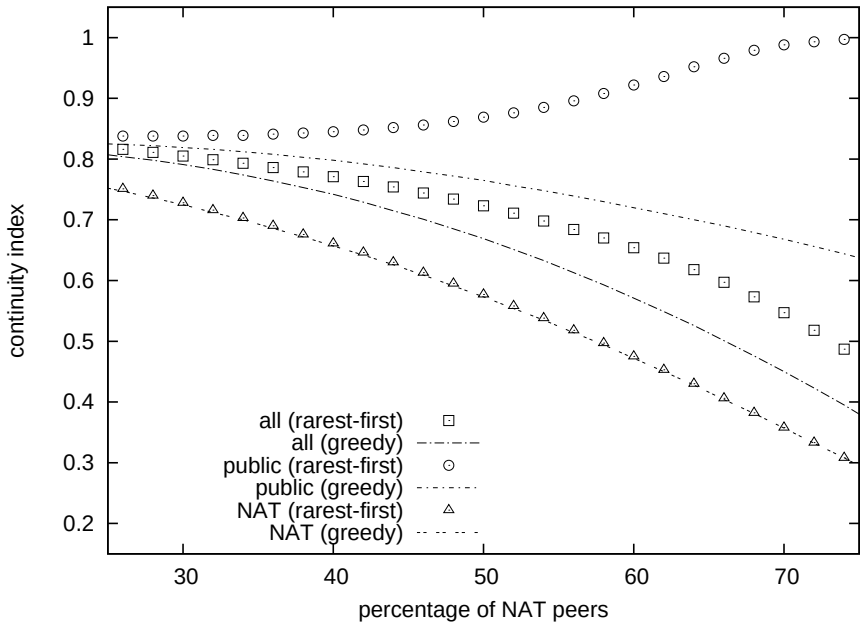

Figure 6.1: Continuity Index (greedy chunk selection)

rarest-first chunk selection already shown in Chapter 4, and the lines are the new results for greedy chunk selection. In general, the video performance of the greedy scheme is worse than that of the rarest-first scheme, which is the same argument in [37] for large buffer sizes (120 chunks in this case). NAT has similar negative impacts on the video performance, no matter which chunk selection strategy is used. However, a major difference is that public peers do not take advantage over the existence of NAT when using the greedy scheme.

This difference is resulted from the way each strategy behaves in the context of the video streaming buffer (see Figure 3.1). In essence, the greedy strategy always tries to download the chunks in the near-end from the playback point in the buffer, while the rarest-first strategy downloads those in the far-end first. From a public peer's point of view, when a video chunk near its playback deadline is missing from the buffer, the greedy strategy will request for this chunk from a selected neighbor. If the selected neighbor does have this particular chunk, it will be downloaded; otherwise it will miss

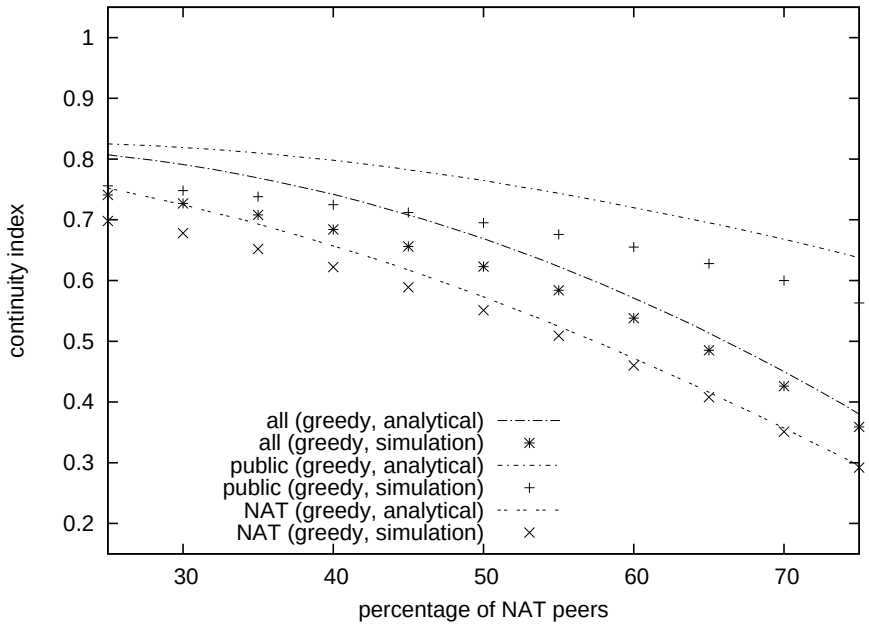

Figure 6.2: Continuity Index (greedy chunk selection, simulation)

the deadline. Therefore, the greedy strategy indicates that a public peer has similar probabilities of having those video chunks in the near-end from the playback point in the video buffer as a selected (either public or NAT) neighbor. Since the continuity index is merely the probability of having the chunk closest to the playback, the continuity index curve of public peers shows the same trend as the overall continuity, which is decreasing as α increases, in Figure 6.1. On the other hand, if the rarest-first strategy is used, similar probabilities of having those chunks in the far-end from the playback point will be seen, but this does not necessarily mean the same trend in the continuity index curves, due to the data exchange of the chunks between the far-end and the playback point. The above explanations on this interesting observation in the preliminary results on the greedy strategy need further investigation and verification.

This observation is also seen in simulation results (Figure 6.2), which validates the analytical results using the extended model. The gap between the two sets of results is slightly larger than that in rarest-first strategy (Figure 4.4). This is because the

measurement in simulation can be skewed by the factor of startup delay, as there is always a trade-off between video continuity and startup delay and the greedy strategy targets at improving the latter.

6.1.2 Heterogeneous Network Bandwidth

Network users behind NAT devices are most commonly seen in home and small office network settings. In most cases, they are sharing a public IP address with other users behind the same NAT device, as well as their total network bandwidth. Therefore, it is reasonable to assume that peers behind NAT devices in a peer-to-peer system often have lower upload and download bandwidth available than their counterparts on the public network.

In the analytical model presented earlier in Chapter 3, it is assumed that all peers have the same uplink and downlink capacity, U and D. In other words, the bandwidth of each peer is *homogeneous*. If we want to address the lower bandwidth for NAT peers, a *heterogeneous* network bandwidth scenario needs to be created, i.e., public peers have a higher uplink and downlink capacity, U_p and D_p, while NAT peers use different values U_n and D_n, which are smaller than U_p and D_p, respectively.

Just like with the greedy chunk selection strategy, the model can be easily extended for heterogeneous network bandwidth as well. All occurrences of U and D need to be examined to determine whether they are the bandwidth for public peers or NAT peers, and then they are replaced by U_p or U_n, D_p or D_n, accordingly.

Since the heterogeneous network bandwidth scenario is only used as another example of extending the analytical model, again, only preliminary results are presented. The symmetric bandwidth case used in earlier chapters is modified and studied. In the modified case, the upload capacity and download capacity of NAT peers are both reduced to 1, while public peers are unchanged. Using the new symbols introduced

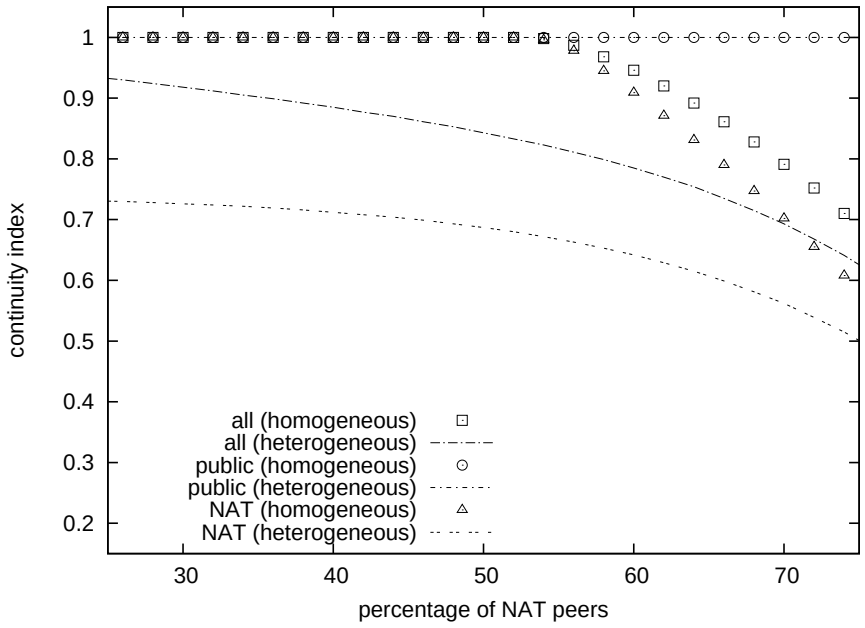

Figure 6.3: Continuity Index (heterogeneous bandwidth)

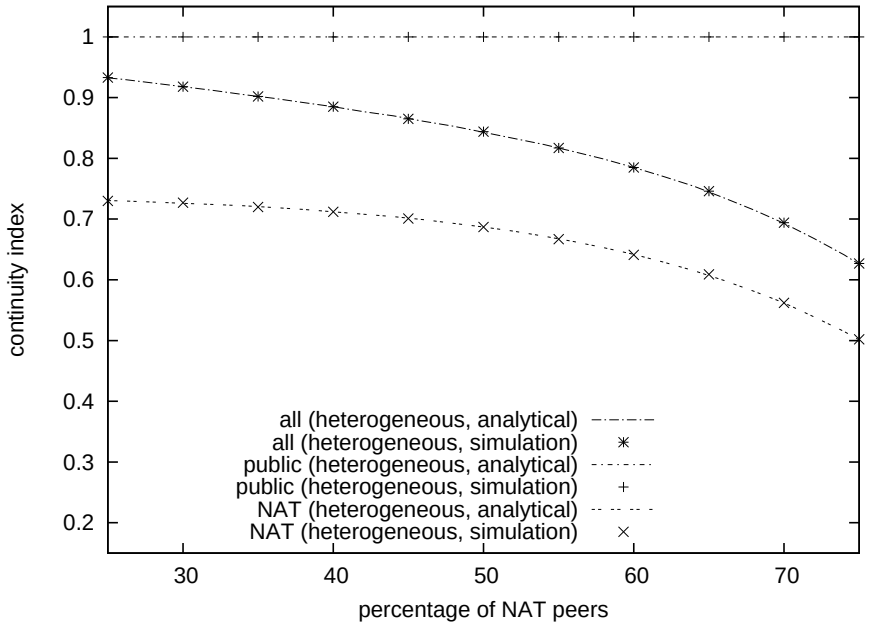

Figure 6.4: Continuity Index (heterogeneous bandwidth, simulation)

for heterogeneous bandwidth, $U_p = D_p = 2$ and $U_n = D_n = 1$. Similar to the first example used in this section, analytical results from the original model are plotted in points and new results from the modified analytical model in lines. Continuity index results are shown in Figure 6.3.

By comparing the two sets of results, it can be observed that the performance of the entire system is even worse in the heterogeneous bandwidth case, especially for NAT peers, which is also seen in the results from software simulation (Figure 6.4). The reason is that NAT peers now have even lower bandwidth to download from other peers, and they do not have enough upload bandwidth to contribute to the system even though they have available video chunks to share.

6.2 Gossip-Based Membership Protocol

In the original CoolStreaming design [2], the client software of each peer has a membership manager module which maintains a partial view of all participating peers in the system, and the partial view is updated by exchanging membership messages. The membership messages are disseminated using Scalable Gossip Membership (SCAM) protocol [18].

The gossip-based membership protocol has its pros and cons. The gossip-based approach requires no centralized node at all, and no node needs to have the global view of all participating peers, therefore, the system is more resilient to single node failures. However, the gossip protocol introduces extra computational and network overhead to the system, and the convergence time is relatively larger since it typically involves multiple hops of message forwarding, when compared with tracker-based systems which only requires one pair of request-response messages.

As an extension of this thesis work, CoolStreaming and gossip-based membership

protocol are studied by means of software simulation. As the software simulation tool described in Chapter 4 is not suitable for simulating the gossip protocol, a new discrete event-driven asynchronous simulator is built for this specific purpose. This simulator is much more realistic and has more randomness, at the expense of a longer simulation time. The new simulator only captures the membership features of a peer-to-peer system, especially the gossiping part, ignoring all video streaming details.

The simulator is inspired by a Google code project called *a new implementation of CoolStreaming with algorithm modification* [50]. This project is also known as *MinaSim* since it is built on the Apache MINA framework [51], which is a network application framework providing an event-driven asynchronous API over various transport layer protocols via Java NIO.

Initially, the gossip simulator implements the SCAMP protocol, strictly following the descriptions in the SCAMP paper [18]. Simulation results show that the average neighborhood size is a logarithm function of the total number of nodes in the system even if no node has any global information about the size of the system. This is also the main argument of the authors of [18], proven by their analytical and simulation results, and thus the gossip simulator is verified, before modifications are made to account for CoolStreaming features and NAT factors in the simulation. The gossip simulator implementing SCAMP is then enhanced to implement the membership and partnership mechanisms in CoolStreaming. In addition, NAT factors are added to the simulator as well. A certain number of the nodes are behind NAT devices, so that they can only initiate connections, but not receive incoming connection requests.

The behavior of the simulator and the system parameters are described below. A peer has up to 4 partners (P_SIZE). A membership message announcing the existence of a peer is generated every 10 seconds ($GOSSIP\_TIME$), and the message is sent to each of the peer's partners. When the message is received, it is forwarded

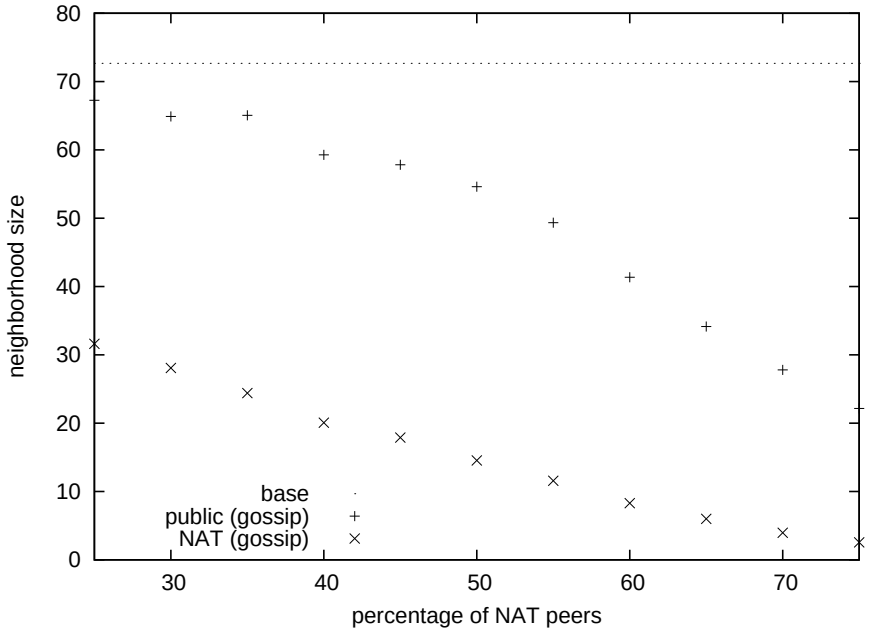

Figure 6.5: Neighborhood Sizes (gossip-based)

to the receiving peer's partners like a gossip. A message can be forwarded up to 3 times ($GOSSIP\_DEGREE$). A peer tries to establish new partnership with peers in its membership every 30 seconds ($ROTATE\_PARTNER\_TIME$). Unlike in CoolStreaming where the rotation is based on a scoring system, the rotation implemented is a mere random rotation, as the video features are intended to be decoupled from the membership simulator. The rotation guarantees that a peer maintains a constant number of partners, and solves the problem which happens when the server returns a NAT peer as the initial contact of a joining peer. In total, there are 200 peers in the system, with a certain percent α of them behind NAT devices. The total simulation time is 30 minutes ($MAX\_SIMULATION\_TIME$).

Note that the above configuration is only for a specific simulation scenario where typical values of the system parameters are used. All the values are configurable to create different scenarios. Only results for this single scenario are presented in this section.

In order to study the impact of NAT on neighborhood sizes in a system using the gossip-based membership protocol, first we need to build a baseline for the system when there are no NAT peers at all. α is set to 0 at first, and the average neighborhood size is measured. Results show that each peer has 72.68 other peers in the system in its partial view on average, for the specific simulation scenario used. After that, more and more NAT peers are gradually introduced to the system by increasing the value of α. The size of membership of each peer is measured, and the average numbers for public peers and NAT peers are calculated separately. Note that a NAT peer in a NAT peer's membership does not count for a member, because they cannot contact each other although they know the existence of one another. A relay mechanism like the one used in [52] is able to establish connections between two nodes behind different NAT devices, but it is not mentioned in CoolStreaming documents and thus not considered in this thesis.

The simulation results are shown in Figure 6.5. The results show that public peers have a much larger membership compared with NAT peers. As the percentage of NAT peer grows, the size of membership for both public and NAT peers drops significantly. In general, a similar trend as that in a tracker-based system is observed (see Figure 4.1), in terms of the impact of the existence of NAT on neighborhood sizes. This indicates that systems using the gossip-based membership protocol do not necessarily work around NAT devices better than tracker-based systems. Recall that Chapter 3 and Chapter 4 have shown that a reduced number of neighbors is a major cause of degraded performance, in terms of both video quality and system fairness. Combined with the results above, it can be inferred that NAT has a huge negative impact on peer-to-peer video streaming systems, no matter whether the gossip-based or tracker-based approach is used for peer list dissemination.

Table 6.1: Reduced Server Bandwidth Usage

	regular			improved		
server	all	pub	nat	all	pub	NAT
100	0.875	0.906	0.803	0.918	0.918	0.917
150	0.913	0.941	0.847	0.951	0.952	0.949
200	0.946	0.969	0.891	0.976	0.976	0.975
250	0.969	0.986	0.930	0.990	0.990	0.990
300	0.985	0.995	0.961	0.997	0.997	0.997
350	0.994	0.999	0.982	0.999	0.999	0.999

6.3 Reducing Server Bandwidth Usage

Previous chapters study a pure P2P system, under both bandwidth-constrained and over-provisioned network conditions. However, the video servers in most systems in the real world not only simply distribute the video contents, but also inject a great amount of upload bandwidth to the system. While the system is bandwidth-constrained from the viewpoint of peers (as their average uplink bandwidth is not much greater than the streaming rate), the overall bandwidth is over-provisioned at the system level, because of the contributions from the video servers.

We have seen in previous chapters that when the server contributions are limited in the system, the existence of NAT devices greatly reduces the system capacity and users experience worse video performance, especially for NAT peers. In order to bring user experience back to an acceptable level, some help from video servers is necessary. This section examines a much more realistic setting (unlike previous chapters which are more focused on pure P2P settings), and answers the questions of how much server bandwidth is needed to make the system sustainable and whether the performance improvement scheme proposed in Chapter 5 can successfully reduce the server bandwidth usage.

In the following simulation, the upload capacity for each peer is equal to the

Table 6.2: Server Bandwidth (95% continuity)

	server	all	pub	nat
regular	210	0.951	0.973	0.900
improved	150	0.951	0.952	0.949

Table 6.3: Server Bandwidth (99% continuity)

	server	all	pub	nat
regular	325	0.990	0.997	0.973
improved	250	0.990	0.990	0.990

streaming rate, but the server capacity is increased to several hundred times higher than the streaming rate. In reality, this is achieved by using a super-high-bandwidth server, or a group of servers. The percentage of NAT peers is set to be 30%, emulating a typical setting where 70% of the video service users are behind NAT devices and about 60% of them can be reached by NAT traversal techniques (suggested by measurement studies [9, 10, 15]). In this experiment, the server capacity is the control variable, increasing from 100 to 350 video chunks per second (equivalent to 50 Mbps to 175 Mbps bit rate if video playback rate is 500 kbps), and the video continuity is measured. Fairness results are not collected as they will be skewed by the large server bandwidth.

Test results are shown in Table 6.1. As expected, the average video continuity is improved as the total server capacity increases. Given the same amount of server bandwidth, the continuity index is considerably higher when the performance improvement scheme is enabled. If an overall video continuity index of 95% is to be achieved, the regular system without performance improvement needs 210 servers (essentially server bandwidth), while only 150 are needed with our performance improvement (see Table 6.2). Similarly, to achieve 99% video continuity, the server bandwidth usage can be reduced from 325 to 250 by enabling the performance im-

Table 6.4: Simulation with Different N and K Values

N, K	$N = 1000$ $K = 50$	$N = 1000$ $K = 70$	$N = 500$ $K = 50$	$N = 500$ $K = 70$
neighborhood size (public)	83.3	96.5	81.3	96.7
neighborhood size (NAT)	35.0	49.0	34.9	48.8
continuity index (all)	0.790	0.806	0.796	0.809
continuity index (public)	0.823	0.837	0.828	0.840
continuity index (NAT)	0.714	0.733	0.721	0.737

provement scheme (see Table 6.3). In these two cases, the server bandwidth is reduced by 28% and 23%, which can significantly reduce the expenses of setting up a peer-to-peer video streaming system providing the same level of service quality.

6.4 Simulation with Different N and K Values

Neither the total number of the peers in the system (N) nor the number of peers returned by tracker (K) is part of the analytical model built in Chapter 3, as they are both canceled out. The software simulation does need both parameters, and it is quite convenient to make modifications to the configuration file and find out if using different N and K values has any effects on simulation results, given that the system is a large-scale system and each peer has enough neighbors (greater than 15).

The simulation results for the bandwidth-constrained scenario with 30% of total peers being NAT peers are summarized in Table 6.4, containing the average neighborhood sizes for public and NAT peers, and the average continuity index for each category, with different combinations of N, K values. The fact that the differences between numbers in the same continuity-related rows of the table are negligible shows that the values of N and K are key factors to neither the analytical model nor the software simulation, which is an argument similar to the analysis in [36].

Data in Table 6.4 also shows that the actual neighborhood sizes for both public

Table 6.5: Biased Tracker ($\alpha = 30\%$, bandwidth-constrained)

scheme	neighbor (public)	neighbor (NAT)	continuity (all)	continuity (public)	continuity (NAT)
none	83.6	35.4	**0.791**	**0.824**	**0.714**
biased peer	83.0	35.2	**0.837**	**0.837**	**0.837**
$K_n = 2K_p$	98.1	70.5	0.827	0.851	0.773
$K_n = 3K_p$	113.1	105.3	**0.836**	**0.844**	**0.819**
$K_n = 5K_p$	142.9	174.7	0.832	0.807	0.891
$K_n = 10K_p$	218.2	350.5	0.796	0.719	0.975
$K_n = 15K_p$	293.2	525.5	0.760	0.660	0.994

Table 6.6: Biased Tracker ($\alpha = 30\%$, over-provisioned)

scheme	neighbor (public)	neighbor (NAT)	share (pub)	share (NAT)
none	83.1	34.9	**1.188**	**0.535**
biased peer	83.2	34.8	**0.996**	**0.986**
$K_n = 2K_p$	98.6	70.1	1.079	0.790
$K_n = 3K_p$	113.2	105.1	**1.000**	**0.946**
$K_n = 5K_p$	143.4	175.4	0.931	1.135
$K_n = 10K_p$	218.9	351.0	0.821	1.384
$K_n = 15K_p$	293.8	525.9	0.760	1.520

peers and NAT peers increase because of an increased K. However, simply increasing the number of neighbors each peer (both public and NAT) has is not a feasible approach to improve the system performance.

6.5 Biased Tracker Mechanism

As briefly discussed in Chapter 5, a *biased tracker* mechanism can help improving system performance by resolving the difference in neighborhood sizes between public peers and NAT peers. The tracker achieves this by sending a list containing the information of K_p peers to public peers while sending K_n ($K_n > K_p$) peers to NAT peers in the list. This mechanism is very straightforward and easy to implement.

Table 6.7: Biased Tracker ($\alpha = 60\%$, bandwidth-constrained)

scheme	neighbor (public)	neighbor (NAT)	continuity (all)	continuity (public)	continuity (NAT)
none	68.9	20.0	**0.642**	**0.898**	**0.471**
biased peer	68.5	19.8	**0.768**	**0.991**	**0.620**
$K_n = 2K_p$	99.0	40.2	0.693	0.968	0.510
$K_n = 3K_p$	128.5	60.2	0.714	0.984	0.534
$K_n = 5K_p$	188.7	99.9	0.731	0.991	0.558
$K_n = 10K_p$	339.0	199.9	0.748	0.993	0.584
$K_n = 15K_p$	489.5	300.3	**0.755**	**0.994**	**0.595**

Table 6.8: Biased Tracker ($\alpha = 60\%$, over-provisioned)

scheme	neighbor (public)	neighbor (NAT)	share (public)	share (NAT)
none	69.6	20.2	**1.673**	**0.457**
biased peer	69.7	20.1	**1.442**	**0.684**
$K_n = 2K_p$	99.7	40.2	1.612	0.530
$K_n = 3K_p$	128.7	60.0	1.584	0.564
$K_n = 5K_p$	189.4	100.2	1.550	0.598
$K_n = 10K_p$	339.7	200.2	1.510	0.633
$K_n = 15K_p$	489.7	300.5	**1.490**	**0.648**

The biased tracker scheme is intended to narrow the gap in neighborhood size between public and NAT peers. Since NAT peers can only have public neighbors, whenever a NAT peer learns about more public peers (from an increased K_n) and connects to them, the neighborhood sizes for those public peers also grow by accepting the connection request from this NAT peer, and there is no way to to increase the neighborhood size for NAT peers only but not for public peers. However, even though both neighborhoods are expanded, it is possible that they are increased at different rates, eventually equalizing the neighborhood sizes (shown in the tables in this section).

In order to show the effectiveness of the biased tracker mechanism, the tracker module of the software simulator is modified to capture the new behavior. While K_p

Table 6.9: Decreasing K_p (bandwidth-constrained)

$\alpha = 60\%$	neighbor (public)	neighbor (NAT)	continuity (all)	continuity (public)	continuity (NAT)
$K_p = 50, K_n = 750$	489.5	300.3	0.755	0.994	0.595
$K_p = 25, K_n = 750$	470.1	300.3	0.762	0.993	0.607

Table 6.10: Decreasing K_p (over-provisioned)

$\alpha = 60\%$	neighbor (public)	neighbor (NAT)	share (public)	share (NAT)
$K_p = 50, K_n = 750$	489.7	300.5	1.490	0.648
$K_p = 25, K_n = 750$	470.0	300.1	1.469	0.664

is kept the default value (50) at first, various K_n values are used to test difference bias levels, from twice the size of K_p to 15 times K_p. The total number of peers N is set to 1000. The simulation results are presented together with those from simulation runs with biased peer selection enabled and without any optimization for comparison purposes. More important performance metrics (continuity index in the bandwidth-constrained scenario and fairness results in the over-provisioned scenario) are highlighted.

Simulation results in Tables 6.5, 6.6, 6.7, and 6.8 show that the biased tracker scheme is able to improve system performance and fairness in a similar way as biased peer selection. An optimal biased level is needed for a certain percentage of NAT peers among all peers. However, when the percentage is high (e.g. 60% as shown), the biased tracker scheme is less effective than biased peer selection, even when a NAT peer almost gets the entire peer population (750 out of 1000) from the tracker. In addition, the neighborhood sizes can be huge for all peers when a biased tracker is utilized, which introduces more network overhead and has more requirement on the computational and storage resources on the client software.

Decreasing the neighborhood size of public peers by returning less peer information

from the tracker can further improve the effectiveness of the biased tracker scheme, but the reduction on extra overhead is limited since the neighborhood sizes are still quite large (as shown in Table 6.9 and Table 6.10).

It is shown that the biased tracker scheme is also effective in improving system performance in the presence of NAT devices. This scheme is easy to implement but it introduces extra overhead to the system. The biased tracker scheme can be used as an alternative or a complement to the biased peer selection scheme proposed in this thesis.

Chapter 7

Conclusions and Future Work

In this thesis, the impact of NAT on the performance of peer-to-peer live video streaming systems is studied. Users of the video service are categorized into public peers and NAT peers, and their performance is examined in both bandwidth-constrained and over-provisioned system settings.

Results show that the presence of NAT degrades the system performance by deteriorating the video continuity and violating the fairness of the entire system. In a bandwidth-constrained system, the video continuity drops significantly as the number of NAT peers increases. While public peers can take advantage of NAT peers, the performance of NAT peers is much worse, which results in a decreased overall continuity index. Adding high-bandwidth video servers to such a system is necessary in order to achieve good user experience. Even if the system is over-provisioned, with a large fraction of the peers being NAT peers, the system is no longer self-sustaining and the performance, especially that of NAT peers, begins to drop. In addition, the fairness issue becomes a major problem in the over-provisioned scenario, as share ratio results suggest that pubic peers are uploading much more than they download, although they can watch the video smoothly.

The analytical model built in this thesis is able to quantitatively analyze such negative impacts, as functions of the percentage of NAT peers in the system. The correctness of the model is supported by software simulation, and it is shown that the model can be easily extended for studying different system variants with minor modifications. The simple performance improvement scheme is shown to be effective in improving both continuity index and share ratio, as well as reducing the usage of server bandwidth. The biased peer selection mechanism can almost completely counteract NAT's negative impacts when less than a half of the peers are NAT peers. With over 50% of the total peers being NAT peers, the scheme still improves the system performance, although the optimal performance cannot be achieved by solely using biased peer selection.

In the future, other performance improvement schemes are to be explored, so that a complement to biased peer selection can be identified. Various system designs on peer-to-peer live video streaming systems can be analyzed using the extended analytical model, so that the impact of NAT on those systems will be better understood. Peer-to-peer Video-on-Demand (VOD) is becoming more and more popular recently. Unlike live streaming, a VOD application user can choose a video program to watch, and each peer contributes disk storage in addition to the uplink bandwidth. It would be interesting to see whether NAT has a similar impact on P2P VOD systems as it does on P2P live streaming systems. It is almost certain that NAT will reduce the system capacity by putting extra restrictions on the connectivity between peers, but its impact on video performance, fairness, or even the storage requirement is to be understood in a quantitative way for P2P VOD systems. Although VOD systems are difficult to analyze because of the lack of synchronization between peers, it is closely related to this thesis, and would make a good follow-on research topic.

Bibliography

[1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proceedings of Workshop on Economics of Peer-to-Peer Systems*, 2003.

[2] X. Zhang, J. Liu, B. Li, and Y. S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2005.

[3] 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 *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[4] P. Srisuresh and K. Egevang, "Traditional IP network address translator (NAT)," *IETF RFC 3022*, 2001.

[5] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session traversal utilities for NAT (STUN)," *IETF RFC 5389*, 2008.

[6] R. Mahy, P. Matthews, and J. Rosenberg, "Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN)," *IETF RFC 5766*, 2010.

[7] J. Rosenberg, "Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols," *IETF RFC 5245*, 2010.

[8] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators," in *Proceedings of USENIX Annual Technical Conference*, 2005.

[9] S. Guha and P. Francis, "Characterization and measurement of TCP traversal through NATs and firewalls," in *Proceedings of ACM Internet Measurement Conference (IMC)*, 2005.

[10] A. Müller, A. Klenk, and G. Carle, "Behavior and classification of NAT devices and implications for NAT traversal," *IEEE Special Issue on Implications and Control of Middleboxes in the Internet*, vol. 22, no. 5, pp. 14–19, 2008.

[11] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer Internet video broadcast," in *(invited) Proceedings of IEEE Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[12] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-VoD system," in *Proceedings of ACM SIGCOMM*, 2008.

[13] A. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *Workshop in Recent Advances in Peer-to-Peer Streaming*, 2006.

[14] H. Chang, S. Jamin, and W. Wang, "Live streaming performance of the Zattoo network," in *ACM SIGCOMM Internet Measurement Conference*, 2009.

[15] S. Xie, G. Y. Keung, and B. Li, "A measurement of a large-scale peer-to-peer live video streaming system," in *Proceedings of IEEE International Conference on Parallel Processing Workshops (ICPPW)*, 2007.

[16] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in *Proceedings of IEEE Global Internet Symposium*, 2006.

[17] C. Dana, D. Li, D. Harrison, and C. N. Chuah, "BASS: BitTorrent assisted streaming system for video-on-demand," in *Proceedings of IEEE Workshop on Multimedia Signal Processing (MMSP)*, 2005.

[18] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 139–149, 2003.

[19] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver-driven mesh-based streaming," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[20] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer live streaming," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[21] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[22] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[23] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[24] PPLive, "http://www.pplive.com/."

[25] PPStream, "http://www.ppstream.com/."

[26] UUSee, "http://www.uusee.com/."

[27] J. Mendes, P. Salvador, and A. Nogueira, "P2P-TV service and user characterization," in *Proceedings of IEEE International Conference on Computer and Information Technology (ICCIT)*, 2010.

[28] P. Srisuresh and M. Holdrege, "IP network address translator (NAT) terminology and considerations," *IETF RFC 2663*, 1999.

[29] S. Guha, Y. Takeda, and P. Francis, "NUTSS: A SIP-based approach to UDP and TCP network connectivity," in *Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, 2004.

[30] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig, "NATBLASTER: Establishing TCP connections between hosts behind NATs," in *Proceedings of ACM SIGCOMM Asia Workshop*, 2005.

[31] F. Audet and C. Jennings, "Network address translation (NAT) behavioral requirements for unicast UDP," *IETF RFC 4787*, 2007.

[32] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh, "NAT behavioral requirements for TCP," *IETF RFC 5382*, 2008.

[33] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, 2007.

[34] L. D'Acunto, J. Pouwelse, and H. Sips, "A measurement of NAT & firewall characteristics in peer to peer systems," in *Proceedings of ASCI Conference*, 2009.

[35] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proceedings of ACM SIGCOMM*, 2004.

[36] S. Tewari and L. Kleinrock, "Analytical model for BitTorrent-based live video streaming," in *Proceedings of IEEE International Workshop on Networking Issues (NIME)*, 2007.

[37] Y. Zhou, D. M. Chiu, and J. C. S. Lui, "A simple model for analyzing P2P streaming protocols," in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2007.

[38] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[39] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in *Proceedings of ACM SIGMETRICS*, 2008.

[40] H. Wang, J. Liu, and K. Xu, "Measurement and enhancement of BitTorrent-based video file swarming," *Peer-to-Peer Networking and Applications (PPNA)*, vol. 3, no. 3, pp. 237–253, 2010.

[41] J. Mol, J. Pouwelse, D. Epema, and H. Sips, "Free-riding, fairness, and firewalls in P2P file-sharing," in *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2008.

[42] J. L. Eppinger, "TCP connections for P2P apps: A software approach to solving the NAT problem," tech. rep., Carnegie Mellon University, 2005.

[43] A. Wacker, G. Schiele, S. Holzapfel, and T. Weis, "A NAT traversal mechanism for peer-to-peer networks," in *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2008.

[44] Y. Liu and J. Pan, "The impact of NAT on BitTorrent-like P2P systems," in *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2009.

[45] L. Chang, Y. Liu, Z. Wei, and J. Pan, "Optimizing BitTorrent-like peer-to-peer systems in the presence of network address translation devices," *Peer-to-Peer Networking and Applications (PPNA)*, vol. 4, no. 3, pp. 274–288, 2010.

[46] L. D'Acunto, M. Meulpolder, R. Rahman, J. Pouwelse, and H. Sips, "Modeling and analyzing the effects of firewalls and NATs in P2P swarming systems," in *Proceedings of IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum (IPDPSW)*, 2010.

[47] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: Optimal performance trade-offs," in *Proceedings of ACM SIGMETRICS*, 2008.

[48] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving a BitTorrent network's performance mechanisms," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[49] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2009.

[50] A new implementation of cool streaming with algorithm modifications - Google Project Hosting, "http://code.google.com/p/coolstreaming/."

[51] Apache MINA, "http://mina.apache.org/."

[52] J. Leitão, R. van Renesse, and L. Rodrigues, "Balancing gossip exchanges in networks with firewalls," in *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS)*, 2010.