Peer-to-peer Stream Merging for Stored Multimedia

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

For the Degree of Master of Science

in the Department of Computer Science

University of Saskatchewan

Saskatoon, Saskatchewan

by

Qing Zhu

Permission To Use

Abstract

In recent years, with the fast development of resource capability of both the Internet and personal computers, multimedia applications like video-on-demand (VOD) streaming have gained dramatic growth and been shown to be potential killer applications in the current and next-generation Internet. Scalable deployment of these applications has become a hot problem area due to the potentially high server and network bandwidth required in these systems.

The conventional approach in a VOD streaming system dedicates a media stream for each client request, which is not scalable in a wide-area delivery system serving potentially very large numbers of clients. Recently, various efficient delivery techniques have been proposed to improve the scalability of VOD delivery systems. One approach is to use a scalable delivery protocol based on multicast, such as periodic broadcast or stream merging. These protocols have been mostly developed for single-server based systems and attempt to have each media stream serve as many clients as possible, so as to minimize the required server and network bandwidth. However, the performance improvements possible with techniques that deliver all streams from a single server are limited, especially regarding the required network bandwidth. Another approach is based on proxy caching and content replication, such as in content delivery networks (CDN). Although this approach is able to effectively distribute load across multiple CDN servers, the cost of this approach may be high.

With the focus on further improving the system efficiency regarding the server and network bandwidth requirement, a new scalable streaming protocol is developed in this work. It adapts a previously proposed technique called hierarchical multicast stream merging (HMSM) to use a peer-to-peer delivery approach. To be more efficient in media delivery, the conventional early merging policy associated with HMSM is extended to be compatible with the peer-to-peer environment, and various peer selection policies are designed for initiation of media streams. The impact of limited peer resource capability is also studied in this work. In the performance study, a number of simulation experiments are conducted to evaluate the performance of the new protocol and various design policies, and promising results are reported.

Acknowledgements

This thesis would not been possible without the support of many people during my Master's study. I would like to take this opportunity to express my appreciation.

First and foremost, I would like to express my sincere thanks to my supervisor, Prof. Derek Eager for his constant support, patience, valuable guidance, and encouragement throughout my research work, for providing me the funding, and for the many hours he devoted to discussing and reading the thesis.

Special thanks go to the members of our networking group, especially Prof. Dwight Makaroff., who advised me and provided feedback on my research work.

My thanks also go to my thesis committee members, Prof. Mark Keil, Prof. Dwight Makaroff, and Prof. Seok-Bum Ko for their valuable advice and suggestions.

I would also like to thank Jan Thompson and other people in the Department of Computer Science for providing necessary assistance on numerous occasions.

Last but not least, I would like to thank my parents for their all-time support, without which I would not be able to finish my work.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AS – Autonomous system

CDN – Content distribution network

CT – Closest target

DNS – Domain name system

DVMRP – Distance vector multicast routing protocol

ERMT – Earliest reachable merge target

GT-ITM – Georgia Tech internetwork topology models

HMSM – Hierarchical multicast stream merging

IP – Internet protocol

LAN – Local area network

MBGP – Multicast border gateway protocol

MSDP – Multicast source discovery protocol

NICE – NICE is the Internet cooperative environment

P2P – Peer to peer

PIM – Protocol independent multicast

PIM-SM – Protocol independent multicast - sparse mode

RTCP – Real-time transport control protocol

RTP – Real-time transport protocol

RTT – Round trip time

VOD – Video on demand

Chapter 1

Introduction

With the development of high speed Internet, multimedia applications are experiencing dramatic growth in recent years. To achieve wide deployment, the technology of *media streaming* is crucial for many multimedia applications, such as distance learning, news-on-demand and movie-on-demand. A media streaming system provides streaming service to remote users at the times of those user's requests. Due to the high bandwidth requirement of media contents, how to provide a *large-scale* service in a media streaming system is a challenging problem that has motivated much research in recent years.

Various multimedia streaming applications are available on the Internet. The two most popular ones are video-on-demand (VOD) streaming and live streaming (e.g., live sports broadcast). This thesis studies the problems in the design of VOD streaming systems. The key challenge is to achieve a *scalable* design that can serve a large number of concurrent clients. Efficiently distributing the media contents so as to reduce the delivery cost is also a concern. This thesis proposes a protocol that addresses these problems, and evaluates the performance. The remainder of this chapter reviews the previously proposed techniques on VOD streaming in Section 1.1, followed by the thesis contributions in Section 1.2 and the organization of this thesis in Section 1.3.

1.1 VOD Streaming Overview

VOD streaming is the *on-demand* distribution of stored multimedia files on the local-area or wide-area networks. Instead of downloading the entire media file before the media playback, a client in a VOD streaming system can immediately begin media playback after the client-side buffer is filled with a small portion of the media file. Typically, the VOD streaming server is the key component which controls the delivery of media files. When the streaming server delivers a media file, the delivery costs are required on both the server end (i.e., the *server bandwidth cost*) and the network channel that is utilized for the delivery (i.e., the *network bandwidth cost*).

Conventionally, *unicast* is used to deliver media files, i.e., there is a distinct connection between the streaming server and each client. However, with the widespread deployment of broadband access, VOD streaming on the Internet has received more and more attention recently. Conventionally, the media streaming is based on the unicast delivery from a single server, which is obviously not practical for wide deployment of VOD streaming service. Thus, for wide-area delivery to a large number of users, many of the recently proposed scalable solutions have employed the technique of *multicast* [4, 39]. By using multicast, a sender can efficiently deliver media files to multiple receivers, resulting in great savings on delivery costs (i.e., both the server and network bandwidth cost).

Currently, the multicast service can be provided in two ways, either as a *network primitive* or as an *application layer* service. Although the network primitive (IP multicast) can achieve the best efficiency in terms of the transfer delay and the network bandwidth usage, there are still many unsolved problems, such as router-state scalability, deployment complexity, and group management. Thus, many researchers have turned to the solutions in the application layer.

Scalable streaming protocols use multicast to serve a large number of client requests for stored video content with a single multicast stream, resulting in great reduction of the required server bandwidth. Two main categories of such protocols can be identified: *periodic broadcast* [1, 23, 26, 30, 36, 37, 44] and *stream merging*

[18, 16, 17]. Periodic broadcast is most appropriate for delivering hot media objects with high request rates. It can scale very well to a large number of clients. However, all clients have to wait a start-up delay before media playback. Stream merging protocols provide immediate VOD service by initiating a new stream in response to each client request. Protocols in this category such as patching [25], hierarchical multicast stream merging [18], and bandwidth skimming [17], apply various techniques to "merge" streams delivering data from the same media file, resulting in reasonably good efficiency regarding the required server bandwidth.

Another approach to address the scalability issue in VOD streaming systems is the use of *content distribution networks* (CDNs). In this approach, the video files are stored in CDN servers which are strategically placed across the Internet, and clients can retrieve the desired content from a close location. This approach can substantially reduce the network bandwidth usage. However, there are still some issues that need to be addressed, such as content placement and load balancing. In addition, since many CDN servers are used, scattered across the Internet, the deployment cost also needs to be considered.

In recent years, with the increasing power of home machines, the *peer-to-peer* (P2P) paradigm has gained tremendous attention. P2P based approaches have been extensively applied in various research areas. Several previously proposed techniques on VOD streaming have been adapted for use with the peer-to-peer paradigm. One example is P2Cast [21] which incorporates the patching technique into a peer-to-peer system to achieve scalable VOD service. However, the patching technique is not considered to be the most efficient technique for scalable VOD.

1.2 Contributions

This thesis studies the problem of scalable VOD media delivery on large network systems, addressing several performance issues regarding the efficient delivery in terms of the resource usage in media delivery. The main contributions of the thesis are:

- A new scalable protocol for VOD streaming is proposed. It adapts a stream merging technique (HMSM [18]), considered to be one of the most efficient in terms of the required server and network bandwidth, to use a peer-to-peer delivery approach. Given that more and more network users are equipped with broadband network access (e.g., cable and DSL) in recent years, this protocol assumes users that are active in the system (i.e., peers) have certain resource capabilities such as outbound bandwidth and buffer space, that allows them to participate in media delivery. This approach undoubtedly reduces the burden at the server side which is entirely responsible for delivering media streams in conventional server-based approaches, so that the server bandwidth cost can be greatly reduced. Meanwhile, the network bandwidth cost can also be substantially saved if some efficient policies are used in media delivery.

- Several protocol policies are proposed that aim to achieve more efficient media delivery in terms of the network resource usage. These include a number of peer selection policies and an extended stream merging policy. The peer selection policies proposed in this thesis mostly concern the saving on network bandwidth cost compared to conventional server-based approaches (e.g., HMSM). In particular, the *location-first* policy chooses the closest peer to deliver streams for new arrivals, while the *location-bandwidth ratio* policy considers both the network bandwidth cost and the workload balance on delivering streams. The stream merging policy proposed in this thesis is extended from the conventional early merging policies [16]. It attempts to achieve greater network bandwidth efficiency by choosing a merge target that is delivered by a close peer, or even not choosing any merge target if no close merge target exists.

- The impact of peer resource limitations and peer heterogeneity is studied in this thesis. A typical peer in a real system is expected to have less outbound bandwidth capacity and less available buffer space compared with a dedicated server in a server-based system. Such limitation could result in significant performance degradation if the protocol is not carefully designed. With the investigation on this impact, this thesis attempts to find the basic requirement of these peer resource capabilities that enables the new protocol to achieve reasonably good performance.

- A scheme of stream failure recovery is proposed. An important peer characteristic in many peer-to-peer systems is that peers frequently join and leave the system, which may make the system unstable and lead to resource wastage. The thesis studies how this peer characteristic will affect the performance of the new protocol. The proposed failure recovery scheme is designed to cope with stream failures caused by any type of peer departure, and attempts to reduce the delivery cost waste to an acceptable level.

1.3 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 reviews previous research related to scalable media streaming. The detailed design of the new protocol is presented in Chapter 3. Chapter 4 describes the methodology for evaluating the performance of the new protocol and presents the simulation results. A summary of the thesis and a discussion of possible future work are presented in Chapter 5.

Chapter 2

Background

With the dramatic growth of resource capability of both the Internet and personal computers, media streaming applications such as *video-on-demand* (VOD) are emerging as potential killer applications in the current and next-generation Internet. However, the problem of providing streaming service in a scalable way is not trivial to solve as videos in such applications are typically bandwidth-intensive. Two types of techniques that have been previously applied to address this problem are:

- Multicast: The technique of multicast can greatly reduce the cost of delivering a video to multiple clients, compared to with conventional unicast delivery.
- Scalable streaming technique: Scalable streaming techniques use multicast together with client grouping and scheduling techniques that allow a group of clients that request the same video to be served by a single multicast stream, and/or content replication at multiple servers or peers, so as to ensure that large numbers of clients can be served without any one system component becoming overloaded.

The remainder of this chapter discusses previous research on the above two techniques. Section 2.1 presents an overview of multicast and its network and application layer implementation. Section 2.2 briefly discusses some previously proposed protocols for scalable media streaming.

2.1 Multicast Service

Traditionally, content delivery applications use the technique of *unicast* that involves a single source and a single receiver. With the fast development of the Internet, more and more applications such as bulk data transfer and streaming media delivery require the delivery of data from one or more senders to a group of receivers (i.e., one-to-many or many-to-many data delivery). When using unicast delivery, these applications need to create a point-to-point connection from a sender to each receiver, resulting in low efficiency as the same data must be sent multiple times. To overcome this inefficiency, the *multicast* service model enables a single transmission to have many receivers.

To support the multicast service model, data with multiple intended receivers must be duplicated somewhere in the delivery path at either network routers or other end hosts. Data delivery tree or mesh structure(s) are constructed, such that only one copy of each data item is sent on each link in the structure, in general serving multiple receivers.

Currently, there are two ways to implement the multicast service: *network layer multicast* and *application layer multicast*. The following two sections give a brief introduction to these two approaches.

2.1.1 Network Layer Multicast

When multicast was first proposed, the functionalities of multicast service were intended to be implemented in the network layer. This is so-called *IP multicast*. In *IP multicast*, data is replicated at the network routers. All receivers join a multicast group which is identified by a class-D IP address. One property of the model is that any node can act as a sender, even a node that has not joined the multicast group. Note that routers and the members of a multicast group must be multicast-capable (i.e., have implementations of the appropriate protocols), although "tunneling" [32] may be used to route across networks that are not multicast-capable.

Currently, *IP multicast* routing protocols (i.e., protocols for establishing the router state that enables multicast data to be correctly forwarded to all receivers) can be classified as *intra-domain* or *inter-domain*. An Internet Autonomous System (AS), or domain, is a collection of IP networks that are under the same administrative control and run the same routing protocols (unicast and multicast). *Intra-domain* protocols operate within a single AS. There are several intra-domain multicast protocols that have been proposed. DVMRP [45] is the first multicast routing protocol that was proposed and uses reverse path forwarding and pruning. The basic idea behind reverse path forwarding is that a router forwards a received multicast packet on all of its outgoing links (except the one on which it was received), only if the packet was received on the link on the shortest path back to the sender. If a router has no attached hosts that have joined the multicast group, pruning is applied to inform its upstream router to stop forwarding multicast packets. Currently, the most widely used *intra-domain* multicast routing protocol is PIM-SM [13] which uses rendezvous point(s) to establish either *group-shared* trees or *source-specific* trees. A group-shared tree is a routing tree that is used by all senders in a multicast group to forward their packets to the receivers. With source-specific trees, a distinct tree is constructed for each sender. Depending on the source traffic concentration, PIM is able to switch from a group-shared tree to a source-specific tree.

To support multicast across different domains, two *inter-domain* multicast protocols have been deployed, namely MSDP and MBGP. MSDP is used by a domain to announce the existence of active sources to other domains, while MBGP enables exchange of multicast route information between different domains. These two inter-domain protocols together with PIM-SM provide an Internet-wide multicast capability, although they are recognized as not providing a truly scalable solution [4]. The reader is referred to previous reviews on Internet multicast evolution [4] and multicast routing protocols [31, 39] for more details.

Although many years of research have been invested on IP multicast, there are still a number of problems with the deployed protocols, in such areas as multicast group management and multicast address allocation, as well as unresolved issues in areas such as multicast traffic pricing, which have seriously affected the extent of deployment and use of IP multicast.

2.1.2 Application Layer Multicast

Recently, as motivated by the problems with IP multicast, a number of *application layer multicast* protocols have been proposed [6, 7, 9, 10, 28]. In contrast to IP multicast, the task of replicating and forwarding packets so as to achieve delivery to multiple receivers is performed at the application layer. The nodes at which application layer multicast support is implemented form an overlay network, in which the links connecting nodes may span multiple links in the underlying physical network. A node can be an end host which is a consumer and/or producer of the multicast data, or a server within an application layer multicast support infrastructure which acts as an application layer router.

Compared with IP multicast, *application layer multicast* may be easier to implement and manage, at the cost of poorer delivery efficiency including higher latency and greater bandwidth usage. How to minimize these inefficiencies is a key issue in designing a *scalable* application layer multicast protocol. To some extent, minimizing delivery latency and minimizing bandwidth usage may be conflicting goals, with the appropriate balance between them being application-specific. For example, for live media delivery, delivery latency may be very important, while bandwidth efficiency is the most important issue when on-demand streaming of stored content is considered.

Unlike IP multicast in which the network routers are typically stable and robust, end-host routers involved in application layer multicast may dynamically join and leave the overlay, requiring corresponding changes to the multicast routing. How to handle changes to the overlay network, both changes to the participating nodes and to the delay and available bandwidth properties of the overlay links, is another important and non-trivial issue in application layer multicast.

Several application layer multicast systems have been proposed in prior work.

Owing to their chosen tradeoffs between the goals of minimizing delivery latency and minimizing bandwidth usage, these systems are somewhat application-specific. Also, note that some of these systems are implemented only at the end hosts while others make use of an application layer multicast support infrastructure. Three systems are briefly reviewed here: Overcast [28], SplitStream [7], and NICE [6].

Overcast [28] is a single-source application layer multicast system designed for delivery of bandwidth-intensive content. Basically, a set of application layer multicast servers are organized into a distribution tree rooted at the source. There are two techniques that are applied to support the high bandwidth requirement. First, the construction of the distribution tree is based on the goal of maximizing each Overcast node's available bandwidth from the source. Second, all Overcast nodes have permanent storage so that the server content can be replicated within the network when the media content is multicast over the distribution tree.

SplitStream [7] is another system based on application layer multicast that seeks to support high bandwidth content distribution. It is built on Scribe [8] to construct and maintain distribution trees. Scribe is an application layer multicast system which is built on top of a structured P2P overlay network called Pastry [38]. The key principle used in SplitStream is cooperative distribution which addresses the problem in traditional tree-based multicast that a small number of interior nodes carry the burden of forwarding the multicast content. Specifically, instead of using a single multicast tree, SplitStream distributes content through a forest of multicast trees. The media content is split into multiple stripes, each distributed in a multicast tree. SplitStream ensures that each node serves as an interior node in only one multicast tree, so that the task of forwarding content is balanced across all participants. In applications where clients need not receive all of the data, such as streaming of layered video, clients participate in only as many multicast trees as their achievable reception bandwidth allows.

NICE [6] arranges the end hosts into a hierarchy of bounded-size clusters and builds a data distribution tree based on this hierarchy. The use of a hierarchy is for the purpose of achieving scalability to large numbers of participants. Most hosts are at the bottom of the hierarchy and maintain state about a number of other hosts, and incur associated control overhead, that is independent of the total number of hosts. However, NICE uses the head of each cluster to send content to its cluster members, so the links close to the head of each cluster may become bottlenecks.

2.2 Scalable Streaming Media Delivery

How to make streaming media delivery more *scalable* is a hot and challenging issue in recent years' research. A number of scalable streaming techniques have been proposed. They can be classified into three categories: *single-server* based, *multi-server* based and *peer-to-peer* based approaches. As suggested by their names, the difference between the first two approaches lies in the number of servers in the media delivery system. In the first approach, it is assumed that there is only one server that delivers all media streams, using multicast. The clients receive streams only from the source server. In the second approach, multiple servers are used to deliver the media content, possibly using adaptations of the scalable delivery techniques originally developed for the single-server context. In the peer-to-peer based approach, media streams are initiated by either the source server or other peers. Peers not only receive streams, but also cache portions of streams and forward cached streams to other peers. This section briefly introduces the techniques applied in these approaches and reviews previous work.

2.2.1 Single-Server Based Approaches

As the name suggests, the server plays an important role in this approach. Typically, the media server has the responsibility of: (1) initiating media streams, and (2) processing media requests. How to efficiently use the server bandwidth is the key issue in making this approach scalable to large numbers of clients. As mentioned above, the technique of multicast is an efficient way to achieve one-to-many delivery service. Thus, protocols in this approach typically deliver each media file using a number of multicast "channels", each corresponding to the server and network resources required to deliver a media stream, using multicast, and, in the case of IP multicast, an IP multicast group. When a client requests a media file, it needs to listen on one or more multicast channels in order to receive that media. Two main classes of protocols that follow this approach are *periodic broadcast* and *stream merging*. Differing techniques are used in each class of protocols to achieve efficiency in terms of the server bandwidth requirement.

One important issue is the *heterogeneity* of the client resource capability such as the client receive bandwidth. One effective scheme to address this issue is *layer-encoded streaming* [34, 43]. In this scheme, the media server transmits each media file in a small number of layers, each associated with a multicast channel. Among these layers, there is a base layer in which a minimal quality encoding of the the media file is transmitted. For successful media playback, a client must at least have receive bandwidth equal to the transmission rate of the base layer, and if there is more bandwidth available, the client can listen to one or more other layers for quality improvement of the media playback.

The following two sub-sections give a brief introduction to some basic techniques using the *single-server* based approach. Specifically, Section 2.2.1.1 introduces *periodic broadcast* and Section 2.2.1.2 describes the technique of *stream merging*. Generally, these techniques can be applied with the incorporation of the layered-encoded streaming scheme mentioned above to accommodate client heterogeneity.

2.2.1.1 Periodic Broadcast

In recent years, various periodic broadcast protocols have been proposed [1, 23, 26, 30, 36, 37, 44], such as the Pyramid Broadcasting protocol, Harmonic Broadcasting, and their variants. For details, the reader is referred to a survey of periodic broadcast protocols by A. Hu [23].

Commonly, periodic broadcast protocols have a *fixed* schedule for transmitting media data and are most useful for distributing *highly popular* media files. In these protocols, the server allocates a fixed number of multicast channels for each media file. Each media file is divided into a number of segments, each of which is *period-*

ically broadcast on an associated multicast channel according to the schedule. The clients receive media data from one or more channels, but need to wait for a period of time termed the start-up delay before beginning playback of the media file. The key advantage of periodic broadcast protocols is that they can scale very well to a large number of clients, since the media files are broadcast on a fixed schedule regardless of the number of client requests received. However, two major drawbacks are suffered in these protocols. One is the unavoidable access latency encountered by all clients due to the fixed schedule, which might be undesirable in some circumstances. The other is the difficulty of providing interactive functions such as fast forward and other VCR type functions.

2.2.1.2 Stream Merging

The protocols in this family try to overcome the problems existing in periodic broadcast protocols by providing *immediate service* to all requesting clients. Once a client sends a request, that client can immediately receive and playback the media. There are various stream merging techniques that have been proposed. Three of them are briefly reviewed in this section, including *patching* [25], *hierarchical multicast stream merging* [18] and *bandwidth skimming* [17].

Commonly in these techniques, the media server initiates multicast streams, each associated with a group of clients. However, they all define their own requirement of the formation of groups. Patching initiates a new media multicast stream delivering the full media file, whenever a client makes a request for the file sufficiently long after the previous request, and the clients arriving close in time will become part of the same group. The other two techniques repeatedly merge all clients that request the same file into larger and larger groups so that the formation of a group follows a hierarchical merging structure. The key difference between hierarchical stream merging and bandwidth skimming concerns their requirement for the client receive bandwidth.

There are various patching schemes that differ in the policy they define to group clients and to initiate a new full-file multicast stream. The most efficient scheme initiates a new full-file multicast stream in response to a client request whenever the time since the previous request for the same file exceeds a *threshold* T. Otherwise, the client listens to the previous full-file multicast stream, and buffers the data received for later playback. Meanwhile, to achieve immediate service, the server initiates a unicast patch stream to that client for the portion missed in the multicast stream. Both the multicast stream and the patch stream are received at the file playback bit rate, so the client receive bandwidth is required to be twice the file playback bit rate. As shown by Eager et al. [18], if the threshold T is chosen optimally, the required server bandwidth increases with the square root of the client request rate.

Hierarchical multicast stream merging (HMSM) [18] is another scalable stream merging technique that greatly reduces the required server and network bandwidth. It requires server bandwidth that increases *logarithmically* with the client request rate. Similar to patching, the most basic form of HMSM requires the client receive bandwidth to be two times the file playback bit rate. Upon the receipt of a file request, the server initiates a new primary multicast stream from the beginning of the file, at the file playback bit rate. Each client listens to its primary stream as well as a secondary stream (merge target) which is a recently initiated ongoing stream, attempting to "catch up" to the clients receiving the earlier stream. When all of the data missed from the secondary stream has been received from the primary stream, the streams (and clients) are said to be "merged", the client continues listening to its secondary stream, and the primary stream terminates. Based on this policy, clients requesting the same file eventually merge into larger and larger groups.

There are various stream merging policies for HMSM that differ in how the secondary stream is chosen [16]. The simplest of them, termed *Closest Target* (CT), simply chooses the closest earlier stream as the merge target. It is not necessarily possible to merge with a merge target computed by CT, since the target stream itself may merge with its own target before it can be merged with by the later stream. Another variant of early merging policy, *Earliest Reachable Merge Target* (ERMT), chooses the closest stream that a client (group of clients) can merge with if no later client(s) merge with the client(s) first.

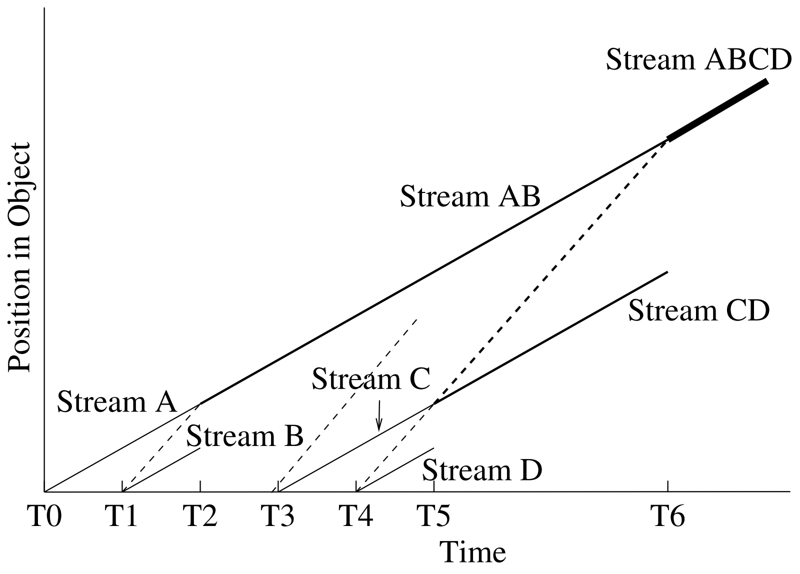


Figure 2.1: Hierarchical Multicast Stream Merging

Figure 2.1 illustrates an example of HMSM using CT. In the figure, four clients A, B, C, and D arrive at times T0, T1, T3, and T4, respectively. Each client is provided with a separate primary stream for immediate playback, as represented by solid lines. Simultaneously, client B, C, and D also listen to the closest previous stream (the merge target secondary stream) until they are merged. The dashed lines show the amount of data that a client (or group of clients) accumulates for an attempted merge. Clients A and B merge at time T2, client C and D merge at time T5, and merged group A and B will merge with group C and D at time T6. As shown for client C, an attempted merge may not occur, i.e., client C merged with client D before it could merge with its initial target which is the group of clients A and B.

The *bandwidth skimming* [17] technique can be considered as a variant of HMSM with a focus on relaying the client receive bandwidth requirement, while still ensuring the bandwidth efficiency achieved in HMSM. It considers the scenario of when a client doesn't have the receive bandwidth of twice the file playback bit rate to perform the hierarchical merging policies described above. The key idea is to hold a portion of the client receive bandwidth that is sufficiently small, to make stream merging workable when the receive bandwidth is only slightly greater than the file playback bit rate.

There are several bandwidth skimming policies defined to perform stream merging with a small "skim" (i.e., difference between the maximum sustainable client receive bandwidth and the file playback bit rate), including *partition*, *latest patch*, *shared latest patch* and *discontinuous patch*. In *partition*, a media file is delivered using K multicast channels, each at a rate of 1/K times the media playback bit rate. Assuming that a client has receive bandwidth equal to (1+1/K) times the media playback bit rate, i.e., it can listen to (K+1) channels, a merge requires K periods, each of the same duration that depends on the time separation between the merger and mergee streams. In first period, the client listens to (K+1) channels including one channel of its merge target stream and K channels of its own stream. In each subsequent period, the client listens to one less channel of its own stream and one more channel of its target stream, until the merge is accomplished and its own stream can be terminated. In *latest patch*, each client listens to a multicast stream delivered at a rate equals to the client receive bandwidth which is greater than the media playback bit rate. When the stream of a client catches up to the playback point of an earlier client, the two clients merge. *Latest patch* has the advantage of low complexity, while *partition* achieves the best efficiency but requires clients to listen to more multicast channels.

2.2.2 Multi-Server Based Approaches

Although the single-server based approaches use various techniques to reduce the required server and network bandwidth, it may not be an optimal solution in terms of the total delivery cost, especially when there are large numbers of clients and a high total client request rate. Previous research on multi-server solutions for scalable streaming media delivery has developed techniques using content replication and proxy caching. Typically, in these multi-server systems, there is a set of replicas or proxy servers storing partial or full media content. Clients communicate with and retrieve the media content from the servers closer to them, so as to reduce the network bandwidth cost.

The conventional multi-server solutions apply the simple unicast streaming for media delivery. This can be further improved if the scalable streaming techniques developed with the single-server based approach are adapted for use with this approach. Several such media distribution systems have been proposed and analyzed [2, 3, 46]. Since it is more complex to design a multi-server media delivery system combined with those scalable streaming techniques, various problems have been studied in terms of how to minimize the total delivery cost, such as replica placement, optimal stream routing and proxy cache allocation. In this section, two of these studies are briefly reviewed.

Almeida et al. study the problem of designing a scalable streaming content delivery system that consists of a number of replicas of popular media content [3]. With some system assumptions (e.g., Poisson client request arrivals), they developed both optimal solution methods and near-optimal heuristics for minimum replica placement and routing. The optimal solution method is applicable with most delivery protocols including stream merging, periodic broadcast, and unicast streaming. Based on the results of experiments with the optimal solution method, they observed that the optimal solution for conventional unicast delivery might be significantly sub-optimal for a system using a scalable delivery technique. The near-optimal heuristics can greatly reduce the execution time so as to support the design of a large and heterogeneous scalable system. These efficient heuristics are based on some good insights of optimal placement and routing: 1) a good placement solution must consider both the network locations and the request rates of the client sites; and 2) the optimal routing involves a tradeoff between minimizing the distance from a replica to each of the clients that it serves, and maximizing path sharing among the clients.

Wang et al. focus on maximizing the delivery cost savings using proxy prefix caching instead of full file replicas [46]. With the system assumption of a source server and a single proxy, the main contributions are the optimal cost model they develop for proxy cache allocation, and various delivery techniques that combine prefix caching with a scalable delivery protocol. The optimal proxy cache allocation scheme is generalized so that it can be applied with any scalable delivery protocol. The delivery techniques that are developed include unicast suffix batching (SBatch), unicast patching with prefix caching (UPatch), multicast patching with prefix caching (MPatch), and multicast merging with prefix caching (MMerge).

2.2.3 Peer-to-Peer Based Approaches

In a peer-to-peer (P2P) system, peers can directly share resources such as files, storage, and CPU capacity with each other. The motivations for use of a P2P-based approach for scalable streaming media delivery include: (1) IP multicast still has unsolved problems related to scalability and deployment; and (2) the multi-server based approaches use a number of dedicated machines, resulting in a relatively large deployment cost.

Various approaches for P2P media streaming have been proposed [5, 11, 14, 15, 21, 22, 29, 42, 49]. Of interest here are those solutions related to *on-demand* streaming of stored media files. Due to the asynchronous nature of client requests in on-demand streaming systems, how to provide scalable delivery service for all requests with synchronous multicast sessions is the key issue in these approaches. A typical technique they apply is called *cache-and-relay*. Specifically, a peer not only receives multicast streams, but also is responsible for caching the received stream and forwarding it to other peers at later time(s). This results in reduction of both the required server bandwidth and the system deployment cost. However, there are several issues that need to be addressed in a peer-to-peer based approach. First, peers normally do not have as much outgoing bandwidth as media servers, so how to select the best forwarding peer must be carefully considered. Second, unlike the multi-server based approaches in which the servers are typically dedicated and robust, peers in peer-to-peer based approaches may join and leave the P2P system frequently. How to handle this dynamic nature is also an important issue. Besides that, the issue of heterogeneity is always important when a widely scalable solution is considered. In this section, three of these approaches, *P2Cast* [21], the *layered peer-to-peer streaming* scheme [11], and an approach that provides near-VOD service [5], are briefly reviewed.

P2Cast [21] incorporates the patching technique into peer-to-peer media streaming to provide scalable VOD service. As mentioned before, the optimal patching scheme aggregates clients that arrive within a threshold period into a multicast session. Each client except the first such client receives two streams initiated by the media server, a full-file ("base") multicast stream, and a unicast patch stream for the missed portion. P2Cast applies the same idea, except that the patch stream can be obtained from a peer rather than the server, while the base stream is still delivered from the server using application-layer multicast. Each peer requesting a particular media file joins the multicast tree for that file to obtain the base stream. Each peer in the tree is responsible for: (1) forwarding the base stream to other peers; and (2) caching its patch stream and relaying to other peers as requested. P2Cast proposed several policies based on the available bandwidth at each peer to define how to select a parent in the multicast tree and a forwarding peer for the patch stream.

The layered peer-to-peer streaming scheme [11] also focuses on the problem of large-scale on-demand media distribution. Besides the issue of asynchrony mentioned above, this scheme also considers the issue of heterogeneity in terms of client resource capabilities. Two techniques are incorporated to address these two issues: *cache-and-relay* and *layered multicast* [34, 43]. In this approach, like previous layer-encoded streaming approaches, a media stream is encoded into multiple layers based on the layer-encoding policy. The server transmits each layer on a separate multicast channel. Each peer receives a number of layers as limited by its inbound bandwidth, and caches a number of layers that may be fewer than it received due to limited buffer space. The outbound bandwidth limits the number of layers that one peer can forward to other peers. Each peer can retrieve layers from more than one peer depending on their request time. The basic algorithm applied in this scheme follows the goal of maximizing the net benefit (i.e., difference between the system benefit and the system cost). The overall streaming quality of all peers is regarded as the system benefit which is defined as the total number of layers received by all peers. The system cost is defined as the server bandwidth consumption.

Annapureddy et al. propose an approach that provides high-quality near-VOD service more practically with P2P technology [5]. Particularly, they investigate the scheduling problem of efficiently disseminating the blocks of a video file in an unstructured P2P mesh-based system. The main contribution of this approach is the various policies proposed for scheduling the propagation of file blocks among peers or from server, with the purpose of ensuring a small start-up delay and maximizing the system resource utilization. Specifically, both the server and client policies have been studied with respect to different scenarios of during or after the initial flash-crowd period. Also, the techniques of pre-fetching and network coding are applied to further improve the system throughput, making efficient near-VOD possible with small start-up delay.

Chapter 3

Peer-to-peer Stream Merging

The key contribution of this thesis is the development and evaluation of a scalable on-demand streaming protocol that delivers media files in a *peer-to-peer* fashion. This protocol employs the previously developed hierarchical stream merging technique to deliver media files on-demand. It can be applied in any *large-scale* network infrastructure that involves heterogeneous clients with various resource capabilities. The main goal of this protocol is to enrich the technique of stream merging by adapting it for the peer-to-peer context, and to thus further improve the efficiency of previously proposed stream merging protocols in terms of the server and network bandwidth requirement.

This chapter presents the method used to implement the stream merging technique in the peer-to-peer environment, and the protocol details. Specifically, Section 3.1 discusses the motivation and outlines the key requirements for developing the new protocol. Section 3.2 presents a detailed protocol overview including the system architecture and some important server and client characteristics. Some design issues such as peer selection policies, impact of limited peer storage capacity, and stream failure recovery are discussed in Sections 3.3, 3.4, and 3.5.

3.1 Motivation

As classified in Chapter 2, previously proposed techniques for scalable streaming media delivery include *single-server* based, *multi-server* based, and *peer-to-peer* based approaches. This section discusses the weakness of these approaches.

Single-server based approaches apply multicast to achieve efficiency in terms of the server and network bandwidth requirement. As there is only one server providing service to all clients, the key focus of these approaches is how to minimize the server bandwidth requirement. In one class of techniques, *periodic broadcast*, a fixed number of media streams are multicast by the server independently of the client request arrivals. In another class of techniques, *stream merging*, the server bandwidth requirement is minimized by aggregating clients into larger and larger groups. In terms of the server bandwidth requirement, these delivery techniques can achieve reasonably good scalability. For example, one of stream merging techniques, HMSM [18], can achieve close to the lower bound of the required server bandwidth. However, the performance improvements possible with techniques that deliver all streams from a single server are limited. In addition, another important scalability metric, the network bandwidth requirement [50], is not the main concern in these approaches. In many circumstances, how to minimize the required network bandwidth in an on-demand streaming system is another key issue in terms of the system scalability.

Multi-server based approaches include those techniques that apply either proxy caches or content replicas to achieve scalable media delivery. Assuming that clients always send their requests to close servers, and that with high probability these servers store the requested file or a portion thereof, substantial savings in delivery cost can be achieved. These approaches consider both the server and network bandwidth requirement, so they may be more efficient solutions than single-server based approaches when the network bandwidth cost dominates the system cost. Conventionally, the multi-server based approaches use unicast for media delivery. To optimize the performance, some recently proposed solutions adapt single-server based scalable delivery techniques for use in multi-server systems. For example, Jussara et al. studied multi-server systems using the stream merging technique [3]. This can further improve the delivery efficiency, but such systems are more complicated than conventional multi-server systems, and in common with other multi-server based approaches, deployment cost is also an issue.

Recently, with the development of broadband Internet and high performance personal computers, some *peer-to-peer* based approaches have been proposed for streaming media delivery. Such approaches have the potential to minimize the deployment cost of a streaming media delivery system, as well as the required server and network bandwidth. Typically, these approaches adapt some previously proposed scalable delivery technique for use in a peer-to-peer system. For example, the patching technique and the layered-multicast scheme are respectively applied in P2Cast [21] and layered peer-to-peer streaming [11]. Of interest here are those scalable techniques that provide immediate service, such as patching and hierarchical multicast stream merging (HMSM) [18]. HMSM out-performs patching with respect to the required server bandwidth, especially at high request rates [18]. In addition, the bandwidth skimming variant of HMSM allows the client receive bandwidth to be less than twice the media playback bit rate, in contrast to patching and basic HMSM [17]. Delivering media content in a peer-to-peer system that employs the HMSM technique to achieve optimal efficiency with respect to both the required server and network bandwidth has not been addressed in prior work.

With the discussion of those previously proposed techniques for scalable streaming media delivery, some key properties desired in the new protocol can be summarized as follows:

- **Scalable**: The new protocol should be applicable in any large-scale networking environment in which there are a large number of heterogeneous clients requesting services.

- **Efficient**: The new protocol should be efficient in both the server and network bandwidth costs. The required server bandwidth should be close to the lower bound of the server bandwidth requirement, and the required network bandwidth should be significantly less than any conventional approach.

- **Deployable**: The new protocol should allow easy deployment and be of minimal deployment cost.

3.2 Protocol Overview

The *peer-to-peer stream merging* protocol proposed in this work aims at implementing the technique of hierarchical multicast stream merging (HMSM) [18] in a peer-to-peer environment. As noted, HMSM has been studied previously only in the single and multi-server contexts. The main goal of the new protocol is to relieve the burden on the server with the assistance of all participating peers, i.e., to adapt the centralized technique to a decentralized environment. Note that, although the new protocol is not a pure peer-to-peer protocol since it requires a media server acting as a control point, all clients in the stream merging system are still referred to as "peers" as in any pure peer-to-peer system. This section presents an overview of the architecture of the new stream merging system and the key techniques applied to achieve the decentralization.

3.2.1 System Architecture

Like other VOD systems, the peer-to-peer stream merging system consists of a media server which stores all the media files, and a set of clients (peers) who may request a media file at any time and from any starting position. As HMSM requires the support of multicast for media delivery, it is assumed that the media server and all the peers in the system are multicast-capable (either the native IP multicast or the application layer multicast).

Basically, while the media server still replies to all peer requests, all participating peers that have requested a particular media file can cooperatively provide assistance in the streaming media delivery using the technique of HMSM. Any appropriate peer is able to be a stream provider and is able to initiate and deliver a media stream to any other peer. An appropriate peer is a peer that has enough bandwidth to deliver a stream to another peer, and enough storage capacity to buffer data after playback so it can be later streamed elsewhere.

One notable characteristic of a peer-to-peer system is that a peer not only receives service, but also provides service. Some recently developed peer-to-peer file sharing systems even restrict the amount of service received according to the amount of service provided. Similarly, in a peer-assisted VOD system, peers provide streams as well as receive streams, typically using a scalable technique called *cache-and-relay* [29], which is also employed in this peer-to-peer stream merging approach. Basically, while receiving a requested stream, a peer also acts as a temporary server, i.e., it caches the received stream in local storage and can later relay the cached stream to other peers that request the same media file.

In the conventional HMSM technique, each requesting client is typically required to listen on two streams, a new *primary* stream and an existing *secondary* stream, each delivered by the media server. The primary stream is provided for the initial portion of the requested file and the secondary stream is actually the merge target (i.e., the stream with which a merger is attempted). Now in the new peer-to-peer stream merging system, with the cache-and-relay technique incorporated into HMSM, those two streams are no longer necessarily delivered by the media server. Instead, each peer caches its received streams for a certain length and can be the stream provider to another peer requesting the same file, as long as the starting point of the requested file is within the cached length.

As mentioned in Section 3.1, the main objective of this peer-to-peer stream merging approach is to further improve the delivery efficiency in a scalable VOD system in terms of the server and network bandwidth requirement, especially the latter. As cache-and-relay requires peers to store the streams they received, any one peer can be a temporary stream provider. If another peer requests the same file later, the media server can choose those existing peers who are geographically close to the new peer, and ask them to relay either the primary or the secondary stream. Thus, the total network bandwidth cost can be reduced. Furthermore, the server bandwidth requirement in the new system is at least as efficient as in the conventional HMSM, which increases logarithmically with the client request rate, since now the task of delivering multicast streams is shared by all participating peers instead of only the media server itself.

3.2.2 Relocating Server Responsibilities

In the conventional HMSM system, the responsibilities of the media server include delivering media streams, processing client requests as well as building and maintaining the hierarchical stream merging structure. The major portion of the server bandwidth is consumed on delivering media data streams, i.e. being the stream provider is the main responsibility of the server. In the peer-to-peer stream merging approach, this is no longer the case, due to the peer-to-peer nature of the new system. In this section, the relocation of server responsibilities in the new approach is explained in more detail.

One key element of the HMSM delivery technique is that clients accumulate data faster than their media playback bit rate by listening on more than one stream, so that they can catch up to earlier clients. In this technique, it is essential for the media server to keep track of the progress of all ongoing streams and meanwhile dynamically maintain the hierarchical stream merging structure. Changes of stream merging structure could result from some basic events such as stream initiations and stream mergers. These events are triggered at the media server by the reception of *join requests* and *merge notifications.*

The *join request* is issued by a client when it requests a particular media file. Upon receiving the message, the server will provide the requesting client two dedicated streams: a primary stream that is newly initiated for the initial portion of the requested file for immediate playback, and a secondary stream (merge target) that is chosen from the existing streams that have been recently initiated for the same file.

The *merge notification* is issued when a merge completion is detected, i.e., an existing client (or group of clients) has accumulated enough data to catch up to an earlier stream. As the media server tracks all active streams and has full knowledge of ongoing mergers, it can be assumed that the merge notification is generated by the server system itself. Upon receiving such a notification, the server will reassign the client(s) a new merge target for the next merger. The client(s) will listen to both the new target stream and the initial target stream. At this time, the initial target stream changes its role for the client(s) to be the primary stream. The initial primary stream will be terminated.

These basic events are the same for all VOD systems that apply the HMSM technique, no matter whether in a conventional client-server approach or a peer-to-peer based approach. However, a key difference between the two approaches lies in the way a primary stream is initiated upon receiving a join request, and who is responsible for initiating this stream in particular.

In the conventional HMSM approach, the media server is the only stream provider in the system. For this reason, processing client requests that requires the server to communicate with the stream initiators is quite straightforward, since the initiator of all streams is the server itself. There is no need for the server to contact any clients for the addition of a new stream or removal of an existing stream. This is not the case, however, in a peer-to-peer based approach. Although the media server could still initiate new streams, in many cases participating peers will be the stream initiators. If an approach in which the media server controls events such as initiating stream and stream mergers is used, communication is required between the media server and peers, whenever a peer is selected to initiate the primary stream or it owns the selected merge target stream. At this point, the media server acts more like a control centre that mainly deals with the control messages and maintains the hierarchical stream merging structure. Note that, to properly perform the peer selection, the media server needs to keep track of the per-client state, which is unnecessary in the conventional approach. This requirement has a negative impact on the scalability of the system. How to address this issue is an important concern.

The following subsections give more details about the server responsibilities mentioned above. Specifically, Section 3.2.2.1 describes how the media server deals with join requests and merge notifications, and Section 3.2.2.2 presents a proposed scheme for maintaining state information concerning the requesting clients, at the media server.

3.2.2.1 Processing Control Messages

The media server controls the initiation of new streams and the merger of any two streams when receiving corresponding control messages. This section presents the detail of this server responsibility in terms of the type of control messages.

Join Request

When a new peer sends a join request, it will be provided the identity of a primary stream and a secondary stream to which it needs to listen. As every existing peer or the server itself could be a stream provider, it is necessary to determine which one should initiate the new primary stream, and which existing stream is chosen to be the secondary stream. Specifically, when the server receives a join request for a particular file, the server will: 1) initiate a new stream if there are no active streams delivering that file, (as in this case, no peer will be able to serve as initiator), or otherwise 2) initiate the selection processes for determining the primary stream initiator and the secondary stream (merge target). In either case, the new peer will be informed of which stream(s) it should listen to, and if applicable, the selected peer initiator will be informed that it should create the primary stream for the requesting peer.

The key issue here is how to select the initiator for the primary stream and which active stream is most appropriate to be the merge target. As will be discussed later, the best choices depend on the objectives (e.g., minimizing network bandwidth and/or balancing peer bandwidth load), and on factors such as the peer locations, and the available peer bandwidth and buffer space.

Merge Notification

A merge notification is issued by the server system indicating the completion of a stream merging operation (a stream merger). It implies that the primary stream associated with a group of peers (merger group) has progressed to the merge point which means the merger peers have accumulated enough data to catch up to another group of peers (mergee group). According to the conventional HMSM technique, when a stream merger is completed, two groups (i.e., a merger group and a mergee group) are merged into a larger group. With a new merge target assigned by the server, the merged group will listen on a new target stream as well as the primary stream of the old mergee group. The primary stream of the old merger group will be terminated.

Stream merging operation is carried out similarly in the peer-to-peer stream merging approach, however it must be defined to be adaptable to the peer-to-peer system in which peers participate in the media delivery. Basically, communication is needed between the media server and a participating peer when a stream merger occurs. The details of how this process is performed are explained as follows:

- Upon receiving a merge notification, the server will initiate a selection process, trying to find a new merge target for the merged group if possible. The merge target selection is based on the merging policies discussed later.

- At the peer side, if the merged group obtains a merge target, it will listen on the new target stream and the stream that was the primary stream of the mergee group as well as the merge target of the merger group; otherwise, the group will listen on the latter stream only. Note that, to the merger group, the initially associated merge target now changes its role to the primary stream of the merger group.

- As the old primary stream of the merger group is no longer useful, it can be terminated. Special consideration is needed when the stream is delivered by a peer. There are two options. One is for the server to inform the peer delivering the stream to terminate it, which involves communication between the server and the peer. Alternatively, if the peer can detect when there are no peers listening to the stream, the peer can decide to terminate the stream.

One point to note here is that the mergee group may have its own merge target when it is caught by the merger group. As required by the merging principles defined in the conventional HMSM, the mergee group may have to alter its merge target and listen on a newly assigned merge target.

3.2.2.2 Maintaining Peer State

One complication in the new protocol is that the media server necessarily maintains state for use in the peer selection process. For the peer selection, the useful state information could include workload, available outbound bandwidth, liveness, etc. Since there might be thousands of peers participating at the same time, precisely maintaining per-client state may result in significant communication overhead. Although this control overhead cannot be completely eliminated, there are many ways to reduce it to an acceptable level to improve the scalability of the new protocol. The scheme proposed in this work is explained as follows.

Basically, scalability is achieved in a similar fashion as in the RTP control protocol (RTCP) [40]. Instead of keeping precise per-client state, the media server maintains an approximate view of the state of all participating peers. The server still updates the peer state upon receiving update messages from peers; however, each peer only sends an update message with a probability p after each time interval of some fixed duration. The value of the probability p is adaptive according to the volume of traffic generated, and is periodically multicast by the media server. With this scheme, during each time interval, only partial peer state will be updated so that the bandwidth consumed on the peer update messages can be controlled to a reasonable level.

3.2.3 Reconsidering Early Merging Policies

In the conventional HMSM approach, as new client requests arrive and stream mergers occur, the media server dynamically builds and maintains a data structure that allows it to determine merge targets. The determination of merge targets follows the early merging heuristic [16]. One should merge the neighbouring streams that can be merged at the earliest point in time. To implement this heuristic, there are two key merging principles followed. First, a merge target should be chosen for a client (or group of clients) when the client's request is first made and when it merges with another group of clients. Second, the client(s) should keep listening to the merge target stream until the merger is successful or another (group of) client(s) preemptively merges with its primary stream. The latter principle implies that a merge target may have to be re-selected for a client (or group of clients) when it is caught by any later clients.

Based on the two merging principles, some efficient early merging policies have been proposed, such as *Earliest Reachable Merge Target* (ERMT) and *Closest Target* (CT), which can be employed by the media server for the determination of a merge target [16]. Note that the definitions of these early merging policies are all time-based; i.e., the progress point in time of each stream is used in determining merge targets. They try to make stream merging events occur as soon as possible with the goal of minimizing the server and network bandwidth requirement.

In order to design merging policies for the peer-to-peer stream context, the two merging principles mentioned above are still followed. However, some modifications need to be made to those conventional early merging policies. To determine a merge target, besides the key parameter which is the progress point in time, other parameters are also considered such as the peer location and the peer bandwidth and storage availability. These parameters are also used for determining a stream initiator for a new request. Using these new parameters, some new policies are defined in this proposed work, as will be described in Section 3.3.

3.2.4 Identifying Peer Characteristics

Redundant servers can be used to address the single point of failure problem in conventional centralized HMSM systems. However, reliability is more difficult to address in the peer-to-peer context, since any peer in the system has the potential to be a temporary server of media streams. The reliability of such a temporary server will significantly affect the performance and reliability of stream delivery, as will its available bandwidth and storage resources. It is essential to understand how the characteristics of peer systems differ from those of a server in a conventional VOD system, if an appropriate system design is to be achieved. In this section, issues concerning peer behaviour and capability with respect to resource heterogeneity and reliability are briefly discussed.

Resource Heterogeneity

When a peer requests a media file, the media server will select one of the peers whose cache still retains the beginning of the media file, if there is at least one such peer, to initiate the primary stream for the newcomer. The peer is selected based on one of the policies that will be discussed in Section 3.3.2.2. For example, to minimize the required network bandwidth cost, the candidate (a peer or the server itself) that is closest to the new peer in terms of the network distance will be selected. However, it is not always the case that the closest peer should be selected due to the nature of heterogeneous peer resource capabilities.

One of the important peer characteristics is the heterogeneity. Peers may be located in different parts of the Internet, may have differing inbound and outbound bandwidth and may have differing storage capacities. To meet the requirement of initiating a media stream, a peer must have available bandwidth sufficient for delivering a stream, and also must have the requested data stored in its buffer. Thus, two important peer parameters are the constrained *peer bandwidth* and the constrained *storage capacity*.

In the conventional stream merging solutions, the client bandwidth is typically referred to as the client receive bandwidth on which there is a corresponding requirement for a specific solution. For example, the most basic HMSM technique [18] requires the client receive bandwidth to be at least two times the media playback bit rate, while bandwidth skimming [17] can reduce this requirement to less than two times the media playback bit rate.

In the peer-to-peer stream merging solution, besides the receive bandwidth, the bandwidth constraints also include constraints on the peer outbound bandwidth and the end-to-end bandwidth. Considering the dynamics of a peer and network connections between peers, note that: 1) the outbound bandwidth of a peer is not stable, since the peer might start a new network application at any time, or an existing application might occupy more outbound bandwidth (e.g., a peer-to-peer file sharing application); and 2) the end-to-end bandwidth available along the path from the providing peer to the requesting peer may dynamically change due to changes in congestion levels or underlying network topology.

The other important parameter related to peer resource capabilities is the storage capacity. In previous stream merging techniques, the local storage is used to buffer data which must be received ahead of its scheduled playback time, so as to enable stream merging. After its playback, the data will be discarded. It is different in the peer-to-peer stream merging system. While a peer still stores data ahead of its playback for later mergers, the data remains in the local buffer even after its playback and no data is discarded until the local buffer is full, which is the basic requirement of the cache-and-relay technique. Note that the larger the local storage capacity that a peer has, the wider the time range of requests that a peer can serve, and the greater the probability that a peer can be served by a close-by peer.

Reliability

As in any other peer-to-peer system, peer reliability is a critical issue. Unlike the media server in a conventional stream merging system which is typically dedicated and static, peers in a peer-to-peer system are often short-lived and their resource capabilities dynamically change. Thus, a peer that is providing one or more streams may terminate an ongoing stream at any time, or severe congestion on the delivery path from that peer may develop. It is essential to develop mechanism to cope with such disruptions.

Disruption in delivery of a stream can be graceful, or unexpected. A graceful disruption occurs when a peer must stop an ongoing streaming session because of a user request to leave the peer group or to terminate the incoming stream that is the source of the outgoing stream (owing to use of a VCR function, for example). An unexpected disruption could be caused by a peer machine crash, underlying network failure (e.g., link breakdown) or available bandwidth fluctuation (at end point or on the path). In any such cases, the listening peer(s) and the media server will not be notified of the stream delivery failure, and thus the listening peer(s) may suffer media playback disruption. In Section 3.5, a recovery scheme is introduced for graceful as well as for unexpected stream delivery disruptions.

3.3 Selection Policies

In the peer-to-peer stream merging system, when a new peer arrives, it typically listens on two streams: a new primary stream and an existing secondary stream (merge target). The media server is responsible for selecting an initiator of the primary stream and selecting the secondary stream from the existing streams. Also, when a stream merging operation occurs, the server needs to select a new merge target stream for the merged group. This section presents some design policies related to these selection processes. Specifically, Section 3.3.1 introduces some policies for selecting a primary stream initiator, and Section 3.3.2 presents policies for selecting a merge target stream.

3.3.1 Selecting Primary Stream Initiator

A new primary stream is, and must be, initiated only when there is a new request for a media file. At this time, the media server is responsible to initiate the process of selecting a stream initiator which could be a peer or the server itself. A peer selection policy is required to determine the most appropriate peer. The selection result depends on the selection policy applied. In all cases, only qualified peers are considered; i.e., peers with sufficient bandwidth and with the required media file data cached locally. The media server stores all the media files and is equipped with the bandwidth required to serve a relatively large number of clients, so it will always be a qualified candidate.

The design of a peer selection policy should be consistent with the main objective which is to minimize the overall delivery cost of the system. In this section, three alternative peer selection policies are described: *location-first*, *location-bandwidth ratio*, and *load-first*. The *location-first* policy attempts to minimize the required network bandwidth, the *location-bandwidth ratio* policy tries to balance the network bandwidth requirement and the delivery workload, and the *load-first* policy considers only workload.

Location-first Policy

As the name suggests, in this policy, the initiator candidate that is closest to the requesting peer in terms of the network distance (i.e., transmission latency) will be chosen. It is reasonably effective since, for a new peer, listening to a primary stream that is delivered by a close one (a peer or the server itself) will be efficient with respect to the network bandwidth cost. The idea is quite similar to that of *content distribution network* (CDN), except that stream initiators are not dedicated and static as CDN servers.

The media server initiates the selection process, as it keeps track of the information of all current peers as well as the associated ongoing streams. The process is started with a set of initiator candidates provided to the requesting peer. The requesting peer can then choose the closest one from the candidate set based on the measurement of the network distance along the path from the peer to those candidates. The determination of the candidate set is straightforward. It simply consists of the server and any peers that are currently receiving the same file, and that are expected to have cached all of the data from the beginning of the file to the current stream position. In large-scale systems, there could be hundreds or even thousands of qualified candidates. It might be impractical to require the requesting peer to evaluate such large number of candidates. To address this issue, the media server may randomly (or using some better method) pick K peers from all qualified candidates, and then the requesting peer need only choose among those K.

The method for measuring network distance is out of the scope of this work, since the performance evaluation of this work will be simulation-based. For estimating network distance in a real system, a number of approaches have been proposed [12, 19, 20, 35]. Based on the constrained peer resource capability, of interest are those approaches that are decentralized and have low processing overhead, such as King and Vivaldi described below.

King, a simple but effective solution proposed by Gummadi et al. [20] does not require any infrastructure deployment or landmark setups as in other solutions [19, 35]. The technique is based on two observations: 1) most end hosts in the Internet are close to their DNS name servers, and 2) recursive DNS queries can be used to measure the latency between pairs of DNS servers. Thus, the latency between two end hosts can be estimated by locating nearby DNS name servers and measuring the latency between them. The latency between two DNS servers is estimated by the difference of two latencies (one end host to two name servers) which can be measured using recursive queries.

Vivaldi [12] is another simple, light-weight approach that applies synthetic coordinate system to accurately predict the communication latency between two hosts. Each host has a coordinate in a synthetic coordinate system such that the distance between two hosts' synthetic coordinates predicts the RTT latency between them in the Internet. This approach is fully distributed, requiring no fixed network infrastructure, and thus fits well in a peer-to-peer system. Specifically, without an explicit measurement, a host can collect latency information from only a few other hosts, compute good coordinates for itself, and then scale its coordinate information to a large number of hosts.

Location-bandwidth Ratio Policy

Both the location of a candidate and the available bandwidth on the path between the candidate and the requesting peer are considered in this policy. For a requesting peer r, the metric used in the selection process is defined as $\mathrm{D}_{ri}/\mathrm{B}_i$, where D_{ri} is the network distance from the requesting peer r to the candidate i, and B_i is the available bandwidth between the candidate i and the requesting peer r (including any constraint imposed by limited outgoing bandwidth at the candidate i). Intuitively, selecting a candidate with larger available bandwidth helps to balance the workload among all participating peers, while a short network distance reduces the network bandwidth consumed in the media delivery. This location-bandwidth ratio policy tries to achieve a good balance between the above two factors.

As in the *location-first* policy, the server can provide a requesting peer with a set of at most K candidates, which the requesting peer can then evaluate. This evaluation requires measurement of the end-to-end available bandwidth between a candidate and the requesting peer, which is constrained by the bottleneck link along the path. Although broadband access becomes more popular in recent years, the bottleneck constraint on the last-mile link is still an issue, even in some broadband networks [33]. Since all the stream providers in the peer-to-peer stream merging system are normal peers (home computers), it is more likely that the end points of the path becomes the bottleneck, especially when the outbound bandwidth is concerned. Thus, in this work, it is assumed that bottleneck links are at end-points, and the end-to-end available bandwidth between the requesting peer and a candidate can then be simplified to be the available outbound bandwidth at the candidate, which is simply the difference of the outbound bandwidth capacity and the outbound bandwidth that has been used for delivering streams.

To more precisely measure the end-to-end available bandwidth in a real system, a number of approaches have been proposed [24, 27, 41]. The basic strategy applied in one of these approaches is using a probe packet stream whose rate is slowly increased [27]. One can determine the available end-to-end bandwidth through observation of the trend of the one-way delay differences of the probe stream packets. If the stream rate is higher than the available bandwidth, then the one-way delays of successive packets at the receiver show an increasing trend, otherwise it will stay unchanged. Thus, by monitoring the transition of the trend, the available end-to-end bandwidth can be determined.

The approach to measuring the end-to-end available bandwidth described above can be modified to be used in the peer-to-peer stream merging protocol, with the attempt to minimize measurement overhead. Specifically, a simplification and two modifications are proposed, for this specific scalable media streaming context. The approach is simplified by requiring each peer to keep track of its available outbound bandwidth. Since it is most likely that the bandwidth bottleneck would be at the end points rather than on the in-between path, if a peer pre-detects a bandwidth bottleneck at its outgoing link, the path measurement will not be initiated. If two end points are not bottlenecked, two modifications can be made to the path measurement, which is similar to the approach in PROMISE [22]. First, the starting rate of sending probe packets can be set equal to the required rate for delivering a media stream, which saves both the processing time and the network resource consumed for transferring unnecessary probe packets. Second, the probe packets can be constructed from the actual media data, which can significantly reduce the communication overhead regarding the network bandwidth usage. Since the evaluation of this work will be based on simulations, these modifications are out of the scope of this work, and will be evaluated in future work.

Load-first Policy

The most appropriate candidate defined in this policy would be the one that is with the least workload regarding the bandwidth contribution to other peers. The contribution workload of a candidate peer is defined as the ratio of the bandwidth consumed by the stream(s) currently being delivered to the available end-to-end bandwidth. This policy is proposed for the two purposes. First, although this policy does not intend to minimize the total network bandwidth cost, it can be a comparison to the performance of the other two peer selection policies. Second, with the focus on the bandwidth load in this policy, the fairness regarding a peer's contribution can be achieved so that a distributed workload among all participating peers can be better balanced.

As in the previous two policies, the selection process is initiated by the server and is accomplished at the requesting peer. The measurement of the contribution load consists of two components: the bandwidth currently consumed on delivering stream(s) and the available end-to-end bandwidth. The first component can be obtained from the candidate peers, which is simply the product of the required rate for delivering a media stream and the number of media streams being delivered. The measurement of the second component (the available end-to-end bandwidth) follows the strategy described in the *location-bandwidth ratio* policy. With the assumption made in that policy, this bandwidth can be simplified to the available outbound bandwidth.

3.3.2 Selecting Merge Target

The process of selecting a merge target is initiated by the media server both when a new request is made and when a stream merger occurs. As in conventional approaches, the determination of a merge target depends on the merging policy employed by the server. However, changes need to be made to the conventional early merging policies to adapt them to the peer-to-peer context. This section presents the details of these changes.

The conventional early merging policies intend to find the earliest possible mergers since these give the earliest reductions in the required server bandwidth. All variants of the early merging policies that were considered in previous work have close to optimal server bandwidth requirement, leaving little room for improvement [16]. However, in the peer-to-peer stream merging system, the determination of a stream initiator and a merge target can have a substantial impact on the network bandwidth requirement. The following design goals with respect to stream merging policies are adapted in this work: 1) the design should follow the basic stream merging principles described in Section 3.2.3; 2) the design should be an adaption of the early merging policies, rather than a replacement; and 3) the design should not be significantly more complex than the existing early merging policies.

The overall stream merging policy defined in the new protocol includes two components: an extended early merging policy and a peer selection policy. The extended early merging policy follows the same basic strategy as the previously proposed early merging policies, and thus it can be considered as a variant in that family. The peer selection policy is a new component, with no analogue in the previously proposed early merging policies. It is motivated by the differing network bandwidth costs of listening to a stream sourced by different peers. The early merging policy extended in this work can be based on any conventional variant in the family of early merging policies (e.g., ERMT and CT).

The remainder of this section is organized as follows. The details of the extended early merging policy regarding the determination of a merge target are presented in Section 3.3.2.1, followed by related peer selection policy given in Section 3.3.2.2. Section 3.3.2.3 describes a special case of selecting a merge target, and a brief summary is given in Section 3.3.2.4.

3.3.2.1 Extended Early Merging Policy

Early merging policies such as ERMT and CT attempt to find the earliest possible mergers. The main difference among the policy variants is with respect to their complexity. These existing variants could be applied in the peer-to-peer stream merging system as well, but this would leave no flexibility with respect to peer selection; i.e., with respect to the goal of choosing a merge target stream that is sourced by a peer that is close in terms of network distance. In this section, a variant of early merging policy is proposed that is compatible with use of a peer selection policy.

The new early merging policy attempts to identify a number of candidate merge targets, rather than only the merge target that is soonest possible to be merged with, which can then be selected among using a peer selection policy. To more precisely define these candidate merge targets, a threshold T is introduced, which represents a time range starting from the current progress point of the merge target computed by a conventional early merging policy such as CT or ERMT. Any active stream whose progress point is located in that range is a candidate merge target. For example, assume that the merge target stream computed by a conventional early merging policy is at a position of 10 minutes into the media file. If the threshold T is set to 20, then any active streams whose current progress point is between 10 and

30 minutes will constitute the candidate merge target set.

Now that the extended early merging policy yields as output a set of candidate merge targets, it is straightforward to generate the merge target that will be finally assigned to a peer (or group of peers), using a peer selection policy. The stream associated with the chosen peer is the final merge target to be assigned. If more than one stream is associated with that peer, the one that would yield the earliest merger is chosen.

Overall, by extending the conventional early merging policies with the capability of generating multiple merge targets, the peer-to-peer stream merging approach is able to determine a merge target using both the early merging policy and the peer selection policy. With this extension, any variant of conventional early merging policies (e.g., ERMT and CT) is appropriate to be applied cooperatively with the peer selection policy. While still aiming at improving the overall delivery efficiency, a balance can be achieved between server/peer and network bandwidth.

3.3.2.2 Peer Selection Policy

A peer selection policy, one component of the new stream merging policy, is required in the process of selecting a merge target when a new client request is made or when a stream merger occurs. In both cases, the associated peer(s) will listen to the new target stream, resulting in change of receivers associated with the merge target stream. Note that the change of receivers will have impact mostly on the network bandwidth cost of a particular stream, rather than the workload of a stream provider. Thus, among the peer selection policies described in Section 3.3.1, the *location-first* policy is the only appropriate choice to be combined with the extended early merging policy, for use in selecting a merge target. With the *location-first* policy, the new stream merging policy is efficient with respect to the network bandwidth consumption, which follows the design goal of the new protocol.

The *location-first* policy is used to select a merge target from the candidate target set generated by the extended early merging policy described in the previous section. For a new peer request, it is straightforward to select a merge target which is simply the stream being delivered by the closest peer to the requesting peer in terms of network distance. However, there is a complication in the case when a merge target is selected for a newly merged group. The issue is how to properly select the closest providing peer for a merged group which may include a large number of peers. To address this issue, it may be feasible to find a particular peer in that merged group to be the representative of the whole group, and then select a merge target for that representative. One option is to randomly select a peer from that group. It is effective since the selection processes performed at the first time when the stream is initiated and at all the subsequent times when other peers joins the group are all based on the *location-first* policy. Although the closest candidate peer to the representative might not be the closest to all the other members in the group, it is definitely a closer one to them. Furthermore, since the selection process is performed at peers, selecting a random peer can avoid substantial workload imbalance.

3.3.2.3 Elaboration on Merge Target Selection

Compared to previously proposed single-server based techniques for scalable streaming media delivery, the new peer-to-peer stream merging protocol attempts to delivery media files more efficiently in terms of the network bandwidth requirement, as described in Section 3.1. This goal can be achieved in the *location-first* policy designed for selecting a primary stream initiator and in the stream merging policy pair used for selecting a merge target. However, based on careful examination of cases in which these policies are applied, it is possible to refine the protocol so as to further reduce the network bandwidth cost.

There is a special case that may have negative impact on the network bandwidth requirement when a merge target is being selected. According to the merging principles in the conventional HMSM technique, which is described in Section 3.2.3, a merge target should be chosen for a peer (or group of peers) when the peer's request is first made and when it merges with another group of peers. Although the extended early merging policy can generate a candidate set of merge targets, the actual number of candidates is still limited due to some restrictions (e.g., threshold

T). When a merge target is finally determined from the candidate set using the peer selection policy, it may be the case that the peer that is delivering the selected merge target is not close to the requesting peer(s). In such case, it becomes an issue whether or not the requesting peer(s) should still listen to a merge target which is delivered by a remote peer (i.e., a peer that is far away from the requesting peer(s)).

It is required in the basic merging principles that a requesting peer or a newly merged group should listen to a merge target no matter where the peer delivering the merge target is located. However, it will obviously influence the network bandwidth efficiency if the merge target is delivered by a remote peer. Note that those basic merging principles were defined in the conventional HMSM technique in which the system is server-based and the server bandwidth cost is of most concern. The conventional HMSM technique attempts to maximally merge streams since the fewer the number of streams being delivered, the lower the server bandwidth cost. Now that the server bandwidth efficiency can simply be achieved by distributing the task of delivering media streams to all participating peers, and the network bandwidth cost is a major concern in the peer-to-peer stream merging approach, the merging principles may be slightly modified to be adaptable for the peer-to-peer context.

Basically, when using the peer selection policy to determine a merge target for the requesting peer(s), if the chosen merge target is delivered by a remote peer, the merge target is simply discarded. In such case, the requesting peer(s) will listen to only the primary stream until the primary stream reaches the end of the media file. Although the modification causes the number of active streams to increase in the new system, higher network bandwidth efficiency can be achieved, and meanwhile, there will be no negative impact on the server bandwidth cost since these active streams are delivered by all participating peers. In this work, threshold D is used for determining whether a merge target selected by a stream merging policy should be discarded. When a merge target is delivered by a peer, whose distance from the representative of the requesting peer(s) exceeds the threshold D, the merge target is then rejected.

3.3.2.4 Summary

By illustrating an example of stream merging policy in the peer-to-peer stream merging technique, a brief summary is given in this section.

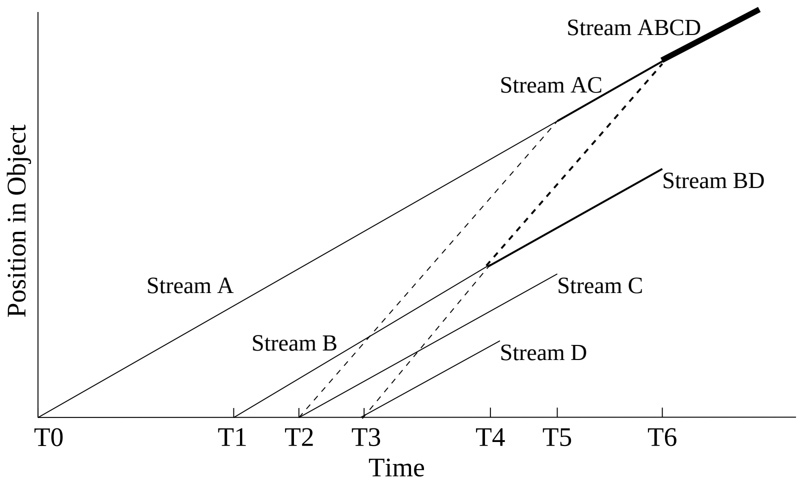


Figure 3.1: Peer-to-peer Hierarchical Multicast Stream Merging

Figure 3.1 illustrates an example of stream merger in the peer-to-peer HMSM technique using stream merging policy pair (location-first policy and extended early merging policy based on CT). In the figure, four peers P_1, P_2, P_3 and P_4 arrive at times T0, T1, T2 and T3, respectively. For immediate playback, each peer is provided with a primary stream, accordingly marked as stream A, B, C and D. These four primary streams are delivered by another four peers P_A, P_B, P_C and P_D, respectively. At the same time, except P_1, the other three peers also listen to a merge target stream for an attempted merger. Each of these three peers should listen to the closest target stream (e.g., peer P_4 arriving at T3 listens to stream C) if the conventional CT policy is used. However, it is different if the stream merging policy pair is used. With an appropriate threshold T, for requesting peer P_4, assume that the extended early merging policy generates two candidate target streams B and C which are delivered by P_B and P_C, respectively. According to the *location-first* policy, if P_B is closer to P_4 than P_C in terms of the network distance, then stream B (delivered by P_B) will be the merge target stream to P_4, which is the case in the example. As shown in the figure, streams B and D merge at time T4, streams A and C merge at time T5, and finally merged stream BD merges with another merged stream AC at time T6. One thing to note is that the attempted merges for some peers may not occur. For example, for P_2, it initially has stream A as the merge target when it arrives. However, this attempted merge is not completed since P_2 will merge with a later arriving peer P_4 before the merge could happen.

As shown in the example, in the peer-to-peer stream merging approach, a stream is not necessarily merged with the earliest stream as does in conventional early merging policies, so it seems not most efficient in terms of the bandwidth consumed at senders. However, as now each peer receives streams delivered by close peers, in terms of the network bandwidth cost, it is more efficient than any conventional approaches. Moreover, instead of the media server delivering all data streams, the task is distributed to all participating peers, thus the server bandwidth efficiency can also be achieved.

3.4 Limited Peer Storage Capacity

So far, the basic techniques of the peer-to-peer stream merging approach and various design policies compared to those in the conventional HMSM technique have been discussed. As mentioned in Section 3.2.4, one of the important peer characteristics is the limited storage capacity. It is essential to study this characteristic in a peer-to-peer based approach. In fact, even in those techniques using proxy caching, the limited storage of the proxy servers is also a key issue so that many researchers have proposed various caching techniques as well as cache replacement schemes. In this section, the impact of this issue on the overall system performance regarding the total network bandwidth cost will be further discussed, and the possibility of achieving improvement upon this limitation will be analyzed.

In the previously proposed stream merging techniques (e.g., HMSM), local storage is used to buffer data that must be received ahead of its scheduled playback time and for all possible mergers. If a client does not have sufficient storage space for a particular merger, the merger is simply not scheduled. The requirement on the client storage capacity has been studied in the approaches using the HMSM technique [18]. It is shown that, to keep the impact on the system performance regarding the required server bandwidth at a fairly small level, a client storage capacity of up to 10% of the file size is sufficient even when the system is under high load.

Compared to the conventional stream merging approaches mentioned above, the peer-to-peer stream merging approach has two differences with respect to the limitation on the peer storage capacity. First, peers are required to have larger storage capacity to achieve the desired system performance. While the same storage space as in conventional approaches is required to accomplish possible stream mergers, a peer needs extra local space to store data even after its playback so that it is able to serve any late-coming peers. Second, instead of having impact on the required server bandwidth, the limited storage capacity in the peer-to-peer stream merging approach will mostly affect the required network bandwidth since the media delivery is accomplished by a set of participating peers all around the network. For example, a requesting peer A asks its closest neighbour peer B for the requested file. If peer B cannot satisfy peer A due to the insufficient local storage space, peer A has to look for another peer who is located further away until it is satisfied, which will obviously increase the network bandwidth cost. Apparently, larger peer storage capacity enables a peer to serve more late-coming peers and to be with greater probability to serve close-by peers, thus resulting in the improvement on the total delivery efficiency. The key point here is how much storage capacity is sufficient for achieving the desired overall system performance, i.e., to find the minimum requirement of the peer storage capacity to achieve the maximum performance gain.

3.5 System Robustness

Another critical issue in a peer-to-peer based approach is the system robustness. Peers are often short-lived, and during a long streaming session either the peer resource capability or network environment on the delivery path dynamically changes. How to handle such peer and network dynamics is an important issue in any peer-to-peer based system. Specifically, as peers are required to participate in stream delivery, in any case of early peer departure, it is essential to accurately and quickly recover a disrupted stream. In this section, a stream recovery scheme that is designed to minimize the impact of stream disruption is presented.

There are two types of early peer departure: graceful departure and unexpected failure. Recovering a stream disruption caused by a graceful peer departure seems not complicated, since the departing peer is able to initiate the stream recovery mechanism before its departure takes real effect. However, recovering a disrupted stream due to a peer failure is not trivial, as there is no prediction of such a stream failure. In order to be consistent for both types of peer departure and meanwhile reduce the complexity of implementation, the stream recovery scheme is designed as a receiver-driven process that is applicable for recovering stream disruption caused by both graceful peer departure and unexpected peer failure.

The stream recovery scheme is based on two components, disruption detection and disruption recovery. The disruption detection process is performed at the receiving peers. Although a disruption event may also be reported by the providing peers in some cases (e.g., graceful departure), it is preferred to be initiated at receivers so that the stream recovery scheme can be applied to both types of peer departure. The disruption detection works as follows. Any receiving peers in the system constantly monitor the incoming traffic including the primary stream and the merge target stream. If the rate of an incoming stream degrades to a certain degree, a stream disruption will be reported. Note the stream disruption in such case could be caused by any reason such as graceful peer departure, peer failure or network fluctuations.

After detecting a stream disruption, the receiving peers report the disruption event to the media server. However, in a real system, there might be more than hundreds or even thousands of receiving peers listening on the disrupted stream. For the server to be notified of the stream disruption, one message will be sufficient.

As recalled in Section 3.2.2.2, a RTCP-like scheme is introduced for maintaining an approximate peer state at the server. In that scheme, peers would periodically send status results with a probability. The status report could include one item of stream status that is used for the stream disruption. Since peers might send their status reports at different time, those whose sending time is coming soon (i.e., sending time – current time $\leq$ interval) will immediately initiate the sending process.

Once the media server is notified of a stream disruption, the stream recovery process is started. Since the disruption of the primary stream will directly affect the media playback, it is quite important to quickly recover the disrupted stream, especially the disrupted primary stream. For this reason, the server will take the responsibility to initiate the recovery stream. Specifically, upon receiving a stream disruption report, the server initiates a new stream starting from the disrupted point, and then notify all the peers who were initially listening on the disrupted stream. Although this scheme may not be the most efficient with respect to the network bandwidth cost, it is chosen as the first choice for continuous media playback.

The recovery scheme described above works for all kinds of stream disruption. However, it cannot completely remove the disruption encountered at peers. In order to minimize the disruption during the media playback, some techniques can be applied to improve the design. Two possible choices are: 1) setting the stream delivery bit rate a little higher than the media playback bit rate, or alternatively, 2) adding an initial startup delay before the media playback. Both choices ensure that more data is received from the primary stream than consumed during the media playback so that more time is allowed to recover the failure. The evaluation of these two techniques is out of the scope of this work, and will be done in future work.

Chapter 4

Protocol Performance

This chapter presents a performance study of the peer-to-peer stream merging protocol based on simulations. The performance of the protocol and the design policies presented in Section 3.3 are evaluated under various network conditions. Section 4.1 presents the goals of the performance study. Section 4.2 describes the design of the simulation study including network topologies, simulation methodology, performance metrics, and experimental parameters. The simulation results are presented and analyzed in Section 4.3.

4.1 Experimental Goals

The peer-to-peer stream merging protocol is developed based on the previously proposed HMSM techniques. It attempts to achieve additional benefit in terms of the network bandwidth cost required in streaming media delivery while still ensuring the server bandwidth efficiency. The main goal of the performance study is to illustrate this benefit over conventional stream merging techniques. Besides that, it also aims at exploring the performance properties of the protocol under a heterogeneous peer-to-peer environment. Specifically, the following questions are considered in this simulation study:

- In terms of the server and network bandwidth requirement, how does the performance of the peer-to-peer stream merging protocol compare to that of the conventional HMSM protocol?

- What is the performance gain of the stream merging policies proposed in this work over the conventional early merge policies?
- How does the performance of the three peer selection policies described in Section 3.3.1 compare?
- How do peer resource capacity limitations on outbound bandwidth and buffer capacity impact the overall system performance?
- How efficient is the failure recovery scheme presented in Section 3.5?

4.2 Experimental Design

Experimental design plays an important role in evaluating network mechanisms and protocols. This section describes the details of the experimental design that is used to evaluate the peer-to-peer stream merging protocol and various related policies. The section is structured as follows. Section 4.2.1 describes how synthetic, representative network topologies are generated for use in the simulation experiments. Section 4.2.2 presents the major part of the experimental design including some system assumptions and the simulation method. Section 4.2.3 defines the performance metrics and related experimental parameters utilized in the simulation.

4.2.1 Network Topology

To simulate activities in various large scale networks, two types of network topology generators are used in this simulation study to obtain representative AS-level and router-level underlying network topologies: GT-ITM [48] and Inet-3.0 [47]. This section provides a brief overview of these two network topology generators.

GT-ITM is one of the most widely-used topology generators for networking research. It can generate both flat random and hierarchical network topologies, typically of tens or hundreds of nodes. While a flat random topology does not reflect the structure of a real internetwork, a topology generated using hierarchical models (e.g., the GT-ITM Transit-Stub model) can model a portion of the real Internet, in which a node represents a router or a switch.

The following explains how a topology is hierarchically constructed in the Transit-Stub model. First, a connected random graph is generated, in which each node represents an entire transit domain. Next, each node in that graph is expanded to another connected random graph, representing the backbone topology of one transit domain. Each node in each transit domain is then attached with a number of random graphs representing stub domains that are connected to that node. Finally, some additional edges are added between pairs of nodes, where a pair of nodes consists of a node from a transit domain and another from a stub domain, or one node from each of two different stub domains.

Inet is another popular network topology generator that constructs models more accurately reflecting the Internet's AS-level structure. In contrast to GT-ITM, it models the entire Internet as a network of inter-connected Autonomous Systems (ASs), and generates larger network topologies with at least 3037 nodes, each representing a single AS.

Inet is a degree-based generator that aims to imitate the connectivity properties of real Internet topologies. The node degree follows power-law distributions. Basically, there are three steps when constructing a topology. First, each of the topology nodes is assigned an outdegree based on power-law distributions. Second, a feasibility test is performed to ensure the resulting topology is a connected network. Finally, the topology is constructed by following three steps: (1) forming a spanning tree using nodes with outdegree of at least two; (2) attaching nodes with degree one to the spanning tree; and (3) according to the pre-assigned outdegree on each node, matching the remaining unfulfilled degrees of all nodes with each other.

In the simulation experiments, various sizes of network topologies are used to simulate either the router-level or the AS-level Internet topologies. For GT-ITM, only the Transit-Stub model is used, yielding topologies varying in size from 25 to 1000 nodes. For Inet, the generated network topologies varies in sizes from 3050 to 7000.

4.2.2 Simulation Methodology

Network topologies generated as described in Section 4.2.1 are used as input to the simulation. Based on the input topology, various events in the peer-to-peer stream merging system can be simulated. This section describes the method used in the simulation for generating various results. So as to simplify the simulation design, the following assumptions are made regarding the network topologies and media delivery:

- Network topologies are modelled in the simulations by undirected and unweighted graphs, with identical unit bandwidth cost on each link.
- It is assumed that shortest-path trees are used for delivering multicast streams. Since any peer or the server could be the root of a multicast delivery tree and the set of peers receiving a stream will generally be different for different streams, a separate shortest path tree is built for each media stream.
- Client accesses to only a single media file are modelled in the simulations. It is assumed that media file data is delivered on each stream at a constant bit rate, equal to the file playback bit rate.
- It is assumed that requests for the media file are generated according to a Poisson process, and that each request is for a full-file playback.
- For the network distance between two peers, the length of the shortest path between them is used, as measured by the number of links.
- For stream delivery, it is assumed that potential bandwidth bottlenecks are at the end-points rather than at interior points along the delivery path, as discussed in Section 3.3.1. The end-to-end available bandwidth between a source peer and a requesting peer can then be simplified to be the available outbound bandwidth at the source peer.
- As long as a peer is active in its uploading session(s), unless stated otherwise, it will stay in the system even after its own receptions complete. However, in

such a case, the peer will not be able to initiate and deliver any new stream to other peers.

- There is an issue in evaluating the media delivery cost. The network topologies generated by the two generators described in Section 4.2.1 are at either the router-level or the AS-level, whereas the peer-to-peer stream merging protocol operates at the host-level. For simplicity, network bandwidth consumption of media delivery among the hosts associated with a single node in the generated topologies is ignored. Specifically, each peer is associated with a network node which is randomly picked when the peer joins the system. There is a possibility that a peer listens to a stream which is delivered by another peer located at the same network node. In such a case, the network bandwidth cost of that particular stream is neglected. In the simulation, only the costs on the links between two routers or two ASs (i.e., two nodes in the topology) are considered.

Based on the above assumptions, the peer-to-peer stream merging protocol was implemented in an event-driven simulator that models a series of events which occur asynchronously in the real stream merging system. Each simulation run consists of a particular combination of a peer selection policy, a stream merging policy, and an input network topology. A peer selection policy is used mostly for selecting stream initiators. With a specific purpose, it could be either of the three peer selection policies defined in Section 3.3.1. For the stream merging policy, either the extended stream merging policy defined Section 3.3.2.1 or a conventional early merging policy could be used in the simulation to determine merge targets. Since the extended stream merging policy is defined on the basis of a conventional early merging policy, it is assumed that the conventional early merging policy used in the simulation is "Earliest Reachable Merge Target" (ERMT) [16]. In those experiments that directly apply a conventional early merging policy for determining merge targets, ERMT is also used for consistency.

Both the media server and each active peer are associated with a network node. As mentioned, each time a file request is generated, the requesting peer is randomly assigned a network node and will be associated with that node until the peer leaves the system. The node corresponding to the media server is picked at the beginning of a simulation run, also at random. For each media stream, a multicast distribution tree is constructed and maintained over the network topology based on the shortest-path algorithm. For the simulation to be more efficient, the shortest path between any two network nodes in a network topology is pre-computed and loaded at the beginning of a simulation run.

After a network topology is loaded, a simulation run starts and the simulator generates a series of system events over time. Each time an event is generated, the type of the event will be determined, which is either a new peer request, a completion of an existing media stream, or a stream merger event. Based on the type of the event, a corresponding change to the delivery system will be performed, as described as follows:

- **New request**: The new peer obtains a new primary stream, and if possible, an existing stream as the merge target. For the primary media stream, a new multicast distribution tree will be created and added to the system. As mentioned, it is a shortest-path tree and rooted at the node with which the selected initiator is associated. For the merge target stream, the only change is to add the new peer to the current multicast group.

- **Stream completion**: A media stream completes when the associated group of peers has received all data of the requested file. At this time, the corresponding multicast distribution tree currently active in the system will be stopped and removed.

- **Stream merger**: A stream merger event causes changes to the distribution trees associated with the two media streams, the merger stream and the mergee stream. Specifically, the merger tree associated with the merger stream will be removed, and the group initially associated with the merger stream will be added to the mergee tree which is associated with the mergee stream.

4.2.3 Metrics and Parameters

In this section, the performance metrics and experimental parameters defined in the simulation study are described in Section 4.2.3.1 and Section 4.2.3.2, respectively.

4.2.3.1 Performance Metrics

The performance metrics demonstrate the effectiveness of various policies proposed in the peer-to-peer stream merging protocol and help in illustrating the performance efficiency in terms of the server and network resource usages over conventional stream merging approaches. The performance metrics defined in this study are summarized as follows:

- Total network bandwidth cost, defined as the total amount of network link bandwidth consumed by stream delivery. It is one of the major costs that need to be concerned in the media delivery, and is essential in evaluating the performance of the new protocol regarding the network efficiency. The measurement of this metric will be given below in this section.
- Total server bandwidth cost, defined as the total amount of server outgoing bandwidth consumed by stream delivery. It is another major cost that can illustrate the performance efficiency on the server requirement compared with the conventional stream merging protocols.
- Total peer bandwidth cost, defined as the sum of outgoing bandwidth consumed at all peers who participate in delivering media streams. This metric can be applied to evaluate the peer participation in the media delivery.

Measurement of Media Delivery Cost

The total media delivery cost considered in this work includes both the server and network bandwidth costs. The total network bandwidth consumed in the delivery system during the simulation is simply the sum of the network bandwidth consumed by each multicast stream. The method used to measure the network bandwidth cost of a particular multicast stream is briefly described below.

It follows the approach proposed by Zhao et al. [50]. Specifically, for a single multicast stream, the total network bandwidth cost can be computed by taking the product of the stream bit rate, the stream duration, and the sum over all links of the multicast distribution tree of the unit bandwidth cost on that link. Since it is assumed that the unit bandwidth cost is identical for all network links, the latter sum is proportional to the total number of the links in the distribution tree. With two previous assumptions of constant stream bit rate (i.e., media playback bit rate) and full-file playback, the total network bandwidth cost of a particular distribution tree can thus be simplified to be proportional to the total number of the links in that tree. As a stream merging event could occur at any time, and to any delivery tree, such that the structure of a distribution tree changes over time and the durations between two changes are variant, the total network bandwidth cost is computed by taking the sum of the network bandwidth cost in each period during which the tree structure remains unchanged. For each period, since the stream rate is constant, the network bandwidth cost can be computed as the period duration times the number of links in the delivery tree.

The total server bandwidth cost during the simulation is simply proportional to the number of streams delivered by the server since the stream delivery rate is constant. As the number of streams delivered by the server changes over time due to the creation of a new stream or the completion of an existing stream, the total server bandwidth cost is computed in the same way as the total network bandwidth cost described above, which is the sum of the server bandwidth cost in each period during which the number of streams delivered by the server remains unchanged.

4.2.3.2 Experimental Parameters

In this section, a number of experimental parameters used in the simulation study are identified. Two types of parameters are summarized in Table 4.1 and Table 4.2, respectively. The parameters shown in Table 4.1 remain constant in all simulation experiments, as described as follows:

Parameters	Constant Value
Stream duration	1
Stream bit rate	1
Unit link bandwidth cost	1
Peer receive bandwidth	2
Threshold T	0.2
Threshold D (GT-ITM)	3
Threshold D (Inet)	2

Table 4.1: Experimental Parameters I

- Stream duration, defined as the time from stream creation to stream completion for a stream delivering the entire file. Since, as described in Section 4.2.2, it is assumed that the stream bit rate is equal to the file playback bit rate, the value of this constant parameter is 1, measured in units of the file playback time.

- Stream bit rate, defined as the bit rate of a stream required for delivering the requested file. The constant value of this parameter is 1, measured in units of the file playback bit rate.

- Unit link bandwidth cost, defined as the unit bandwidth cost on a link in a network topology. Since each link in a network topology is assumed to be identical, constant value of 1 is used for unit link bandwidth cost so that the network bandwidth cost of a stream can be computed by the total number of links in the distribution tree.

- Peer receive bandwidth, defined as the inbound bandwidth capacity that a peer can use for receiving streams, in units of the stream delivery bit rate.

As the inbound bandwidth is not a major concern in this work, each peer is assumed to have sufficient receive bandwidth which is two times the stream bit rate for receiving two streams, the primary stream and the merge target stream.

- Threshold T, defined as the percentage of the full-file stream duration. It is used in the extended stream merging policy to identify the qualified merge targets. As described in Section 3.3.2.1, it represents a time range starting from the current progress point of the merge target computed by a conventional early merging policy (e.g., ERMT). Any active stream whose current progress point is located in that range is qualified to be a candidate of merge target. Obviously, the greater the value of threshold T, the more qualified merge target candidates among which a merge target can be later selected using the location-first policy, i.e., the greater the probability that the selected merge target is delivered by a closer peer. However, with a too large value of T, it is possible to obtain a merge target whose progress point is far from that of the mergee stream, in which case it will take more time to accomplish the merging process. To be relatively consistent with the conventional early merging policies that attempt to make stream merging events occur as soon as possible, a reasonable value of 20% is used for the threshold T in this simulation study.

- Threshold D, defined as the network distance (measured by number of links) beyond which a merge target selected by a stream merging policy should be discarded, as described in Section 3.3.2.3. With the use of this threshold in the extended stream merging policy, the performance in terms of the network bandwidth cost can be improved. Different values of the threshold are used for GT-ITM and Inet topologies, owing to the differing path length distributions in these networks.

The experimental parameters summarized in Table 4.2 have significant impact on the performance of the new protocol. The values of these parameters may vary in some experiments and be constant in the others, depending on the type of the experiment. Thus, as shown in Table 4.2, each parameter has a default value and a range in which its value can vary. In all sets of experiments, if a parameter is not further specified, its default value will be used. The definitions and settings of these parameters are described as follows:

Parameters	Default Value	Range
Network size (GT-ITM) Network size (Inet)	500 4000	25 - 1000 3050 - 7000
File request rate	1000	10 - 2000
Stream merging policy	extended	conventional, extended
Peer selection policy	location-first	three selection policies
Peer outbound bandwidth	unlimited	unlimited, 1 - 10
Peer buffer size	100%	5% - 100%
Peer failure rate	0	0 - 0.8
Heterogeneity	homogeneous	homogeneous, heterogeneous

Table 4.2: Experimental Parameters II

- Network size, defined as the number of nodes in a network topology. For the router-level hierarchical network topology GT-ITM, a range of network sizes from 25 to 100 is chosen. For the larger AS-level network topology Inet, as it generates network topologies with at least 3037 nodes, a value of 3050 is used as minimum. A maximal network size of 7000 is used due to the high system resource requirement of the simulation experiments.

- The file request rate, defined as the number of file requests that arrive in the duration of time required for a full-file playback. Its default value is 1000, and it can vary from 10 to 2000.

- The stream merging policy. A stream merging policy is required in determining a merge target stream. The extended stream merging policy proposed in Section 3.3.2.1 is compared with the conventional early merging policy in the experiments presented in Section 4.3.3. By default, the extended stream merging policy is used in the other experiments.

- Peer selection policy. A peer selection policy is a major part in the simulation for the new protocol to be effective. The performance of three peer selection policies defined in Section 3.3.1 are evaluated in the experiments presented in Section 4.3.2. By default, the location-first policy, one of the core design in this work, is used in the other experiments.

- Peer outbound bandwidth, defined as the outgoing bandwidth capacity that a peer can use for delivering streams, in units of the stream delivery bit rate which is assumed to be equal to the file playback bit rate in this work. Except for the experiments that study the impact of limited peer outbound bandwidth, in which the peer outbound bandwidth varies from 1 to 10, it is unlimited by default in all the other experiments.

- Peer buffer size, defined as the storage capacity that a peer uses to buffer file data, in percentage of the full-file length. As described in Section 3.4, the file data to be cached includes those being read ahead of its playback time so as to enable stream merging, and those after its playback, so that the peer can initiate streams for other peers. The default value of peer buffer size is 100% which can be thought of as unlimited storage capacity. In those experiments that study the impact of limited peer buffer space, the value of this parameter varies from 5% to 100%.

- Peer failure rate, defined as the number of peers that depart early during the simulation time, divided by the number of active peers in the system. Here, an active peer is a peer who is actively receiving or delivering at least one stream. Note that, this parameter is only applicable in the simulation conducted for peer departure at random (Section 4.3.6), in which its value varies from 0 to 0.8.

- Heterogeneity. This parameter is closely related to the peer resource capability including the outbound bandwidth capacity and the storage capacity, which can be homogeneous or heterogeneous. In the simulation experiments, homogeneous peer resource capability is used by default, and in those experiments that assume limited peer outbound bandwidth or limited peer buffer space, the impacts of both homogeneous and heterogeneous peer resource capabilities are studied.

4.3 Experimental Results

A number of experiments were performed in this simulation study. These experiments evaluate the performance of the new protocol including various related design policies, and study the impact of limited peer resource capability as well. This section presents the details of these experiments and analyzes the detailed results obtained from these experiments.

To show the performance in different aspects, each experiment is generally performed by varying one of the experimental parameters shown in Table 4.2 such as the file request rate and the network size. In order to generate more accurate results, each experiment includes multiple number of runs that differentiate at the network topology input. In each simulation run, a different network topology is used for each particular size of network topology. The different network topologies in same size are generated using different random number seeds. If not further specified, 3 simulation runs were performed in those experiments in which the network size varies, and 10 runs in all the other experiments. Typically, the results shown in figures presented in this section are computed by averaging the results generated from multiple runs.

All simulation results presented in this section have 95% confidence intervals that are within ±5% of the reported values. The confidence intervals are computed using the method of batch means which is based on T distribution. Basically, due to the initial transient, a warm-up period of results is rejected from the beginning of a simulation run. The remainder after the warm-up period can be divided into a number of batches. The duration of a batch is long enough so that the batches are approximately independent. The number of batches in a particular run is dynamically calculated. Each time at the end of a batch, the confidence interval up to the current batch is computed using T distribution, and then compared with the expected confidence interval. The simulation is not stopped until the expected confidence interval is satisfied. In all experiments, lengths of 10 and 20 units of full-file playback duration are used for the warm-up period and the batch duration, respectively.

The remainder of this section is organized as follows. Section 4.3.1 compares the performance of the peer-to-peer stream merging protocol with the conventional HMSM protocol. Section 4.3.2 compares the performance of three peer selection policies defined in Section 3.3.1. Section 4.3.3 evaluates the performance of the extended stream merging policy defined in Section 3.3.2.1 with a comparison to a conventional early merging policy. The impacts of limited peer outbound bandwidth and limited peer buffer space are studied in Section 4.3.4 and Section 4.3.5. Finally, the performance of the stream recovery scheme proposed in Section 3.5 is evaluated in Section 4.3.6.

4.3.1 Comparing Peer-to-peer Stream Merging with HMSM

The conventional stream merging approaches (e.g., HMSM) mainly concern the delivery efficiency at the server side. For example, the server bandwidth requirement of HMSM grows only logarithmically as the client request rate increases. Compared to the conventional approaches, the main goal of the new peer-to-peer stream merging approach is to achieve efficient delivery in terms of the network bandwidth requirement while still ensuring the efficiency on the server bandwidth requirement. Before evaluating protocol performance with the limitation of peer resource capability, the new protocol is studied in an ideal environment. Assuming that all peers have unlimited outbound bandwidth and unlimited buffer space, the performance of the new protocol is first evaluated with a comparison to that of a conventional stream merging approach. In this section, the results of these experiments are presented.

The peer-to-peer stream merging protocol consists of two key components: a peer selection policy and a stream merging policy. To achieve the design goal of the new protocol, the basic form of the combination of these two policies would be the location-first policy and the extended stream merging policy, both of which attempt to reduce the network bandwidth cost. Unless stated otherwise, this basic combination is used in the evaluation described in this section, representing the default setting of the peer-to-peer stream merging protocol. Regarding the conventional stream merging approach used for the comparison, the single-server based HMSM approach is applied, in which the media server initiates and delivers all streams, and stream merging operation follows a conventional early merging policy which is ERMT in this work.

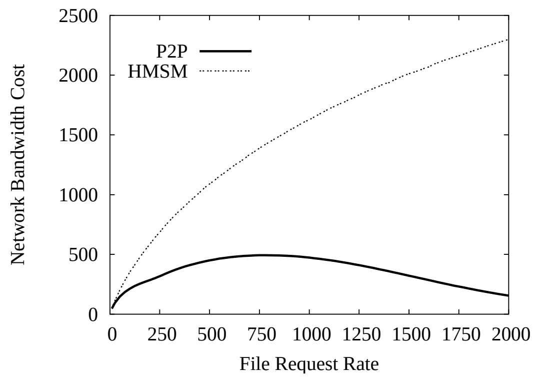
The following two subsections study the protocol performance affected by file request rate and network size, and present the obtained results. Note that, in all figures presented in this section, the lines marked by "P2P" represent the results for the new protocol and those marked by "HMSM" are for the conventional HMSM.

4.3.1.1 Effect of File Request Rate

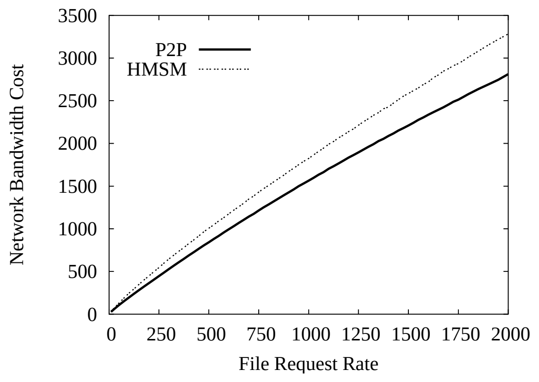
The first set of experiments studies how protocol performance scales with increase in file request rate. Figure 4.1 and Figure 4.4 present the results in terms of the network/server bandwidth cost, and illustrate how efficient the new protocol is compared with the conventional HMSM. Both bandwidth costs shown in figures are assessed as a function of file request rate.

Figure 4.1 provides the network bandwidth costs for the new protocol and the conventional HMSM. For each protocol, the experiments use both network topologies of GT-ITM and Inet, the results of which are respectively illustrated in Figure 4.1(a) and Figure 4.1(b).

Two observations can be obtained from Figure 4.1. First, in both types of network topologies, the network bandwidth cost of HMSM shows an increasing trend as the file request rate increases, and the increasing rates are quite similar. Second, no matter in which type of network topology, the peer-to-peer stream merging protocol always performs better than the conventional HMSM, i.e., less network bandwidth



(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)

Figure 4.1: Effect of File Request Rate on Network Bandwidth Cost: Peer-to-peer Stream Merging vs. HMSM (unlimited peer resource capability)

is consumed in the peer-to-peer stream merging protocol than it is in HMSM.

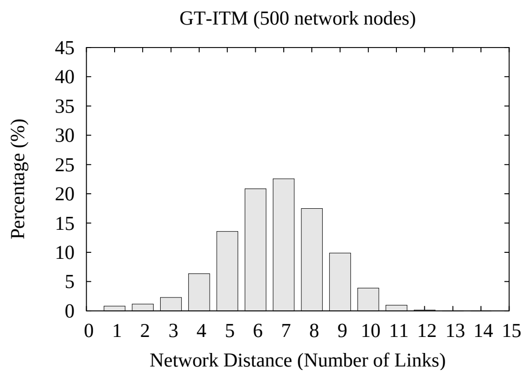
However, there are two variances in performance of peer-to-peer stream merging protocol between two types of network topologies. First, the network bandwidth cost varies quite differently in both network topologies as the file request rate grows. In Inet, like HMSM, it shows an constantly increasing trend, except that the increasing rate is slightly lower than that of HMSM. In GT-ITM, the network bandwidth cost is increasing until it reaches the peak at the file request rate of around 750, and then slowly decreases. Second, compared to HMSM, the amount of network bandwidth that can be saved by the new protocol largely differs in both network topologies. As can be observed from Figure 4.1(a), the network bandwidth cost of the new protocol in GT-ITM is about 50% of that in HMSM at the request rate of 500, and is only about 10% at the request rate of 2000. However, the new protocol saves much less network bandwidth in Inet. As shown in Figure 4.1(b), only about 10% of saving can be achieved even at the request rate of 2000.

The following analyzes some possible reasons for the above performance variances between GT-ITM and Inet topologies.

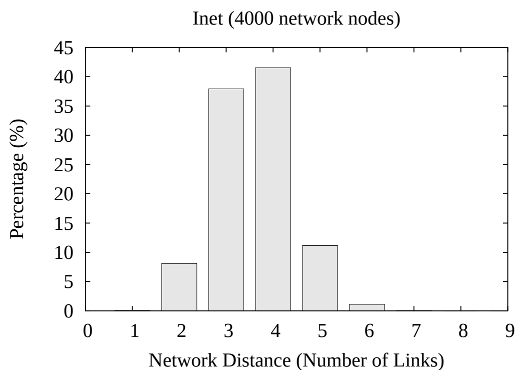
First, the internal structures of GT-ITM and Inet topologies are different. As described in Section 4.2.1, GT-ITM and Inet topologies used in this work reflect router-level and AS-level network structures respectively. The particular Transit-

Stub model of GT-ITM generates topologies with hierarchical internal structure, while Inet generates degree-based network topologies.

To further investigate such difference regarding the nature of internal structure between two types of network topologies, an additional experiment was performed to study the allocation of network distance in both GT-ITM and Inet. As mentioned, both of GT-ITM and Inet are connected topologies, in which each pair of network nodes is associated with a network distance that can be measured as the number of links. Thus, in terms of the value of network distance, there is a distance allocation for all pairs of nodes in a network topology. Figure 4.2 illustrates such distance allocation in a 500-node GT-ITM topology and in a 4000-node Inet topology. In the figure, Y-axis labelled by "Percentage" is expressed as the percentage of two-node pairs among all node pairs at a particular distance. Some observations obtained from the figure are described as follows:



(a) GT-ITM (distance maximum = 14)



(b) Inet (distance maximum = 8)

Figure 4.2: Comparing Network Distance Allocation: GT-ITM vs. Inet

- The ranges of distance allocation in GT-ITM and Inet are different. In Inet, the distances of all node pairs allocate in the range of 1 and 8, while the allocation range is wider in GT-ITM, which is from 1 to 14.

- About 80% of distances in Inet densely distribute at distances of 3 and 4 (38% at distance 3 and 41.5% at distance 4). However, around 80% of node pairs in

GT-ITM are with distances in a larger scatter from distance 5 to 9, and the maximal percentage is only 22.5% at distance 7.

The above observations likely explains why the new protocol yields better performance in GT-ITM than in Inet. Comparing the new protocol to the conventional HMSM, the saving on the network bandwidth cost is minor in Inet, since the distances of most node pairs are close at 3 and 4 as described in the second observation, resulting in no big difference between the network bandwidth consumed by a stream delivered by a close peer as defined in peer-to-peer stream merging, versus that consumed by a stream delivered by the server as required in HMSM. However, with a wider range of distance allocation in GT-ITM, it can save much network bandwidth cost since a peer located far from the server (e.g., with a distance of 8) will listen to a stream delivered by a close peer (e.g., with a distance of 5) in the new protocol, instead of having to listen to a stream delivered by the server in HMSM.

Second, different network sizes of GT-ITM and Inet could also be the reason that causes the performance variance. As shown in Figure 4.1, the default network sizes of GT-ITM and Inet used in experiments are 500 and 4000 respectively. At the same file request rate, it is normal for the same protocol to achieve distinct performance in networks of different sizes. To study how this issue is strongly related to the above performance variance, further investigation was performed as introduced below.

Recall there is an assumption concerning the evaluation of stream delivery cost in Section 4.2.2. In both router-level or AS-level topologies, the network bandwidth cost of media delivery among the hosts associated with a single node is not calculated, since the network bandwidth cost of such stream delivery is neglectable compared to the costs of other deliveries over routers or ASs. Any such stream is referred to as an "no-cost" stream in this simulation work. Apparently, the proportion of such "no-cost" delivery will significantly affect the performance in terms of network bandwidth cost in a simulation run. Figure 4.3 shows the comparison of peer-to-peer stream merging protocol in both GT-ITM and Inet topologies, in terms of the scale of "no-cost percentage" with increase in file request rate. Here,

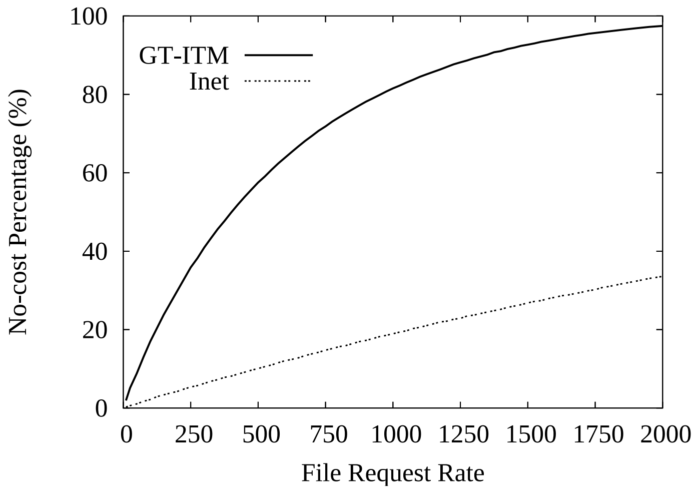


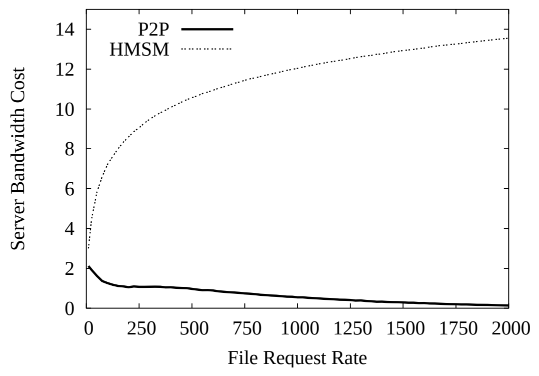
Figure 4.3: Effect of File Request Rate on No-cost Percentage for Peer-to-peer Stream Merging: GT-ITM vs. Inet

"no-cost percentage" is defined as the percentage of new arrivals that listen to a no-cost stream (i.e., the primary stream for a new peer).

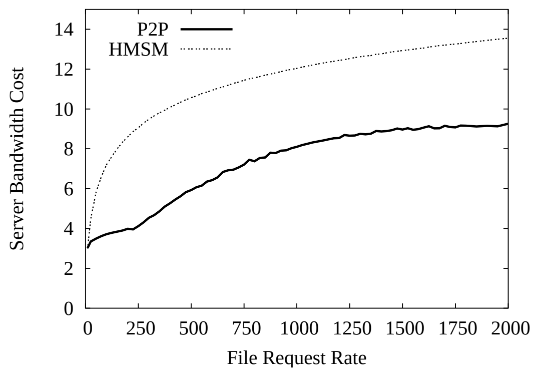
Observe from Figure 4.3 that the no-cost percentage is much higher in GT-ITM than in Inet, which means much more peers in GT-ITM listens to no-cost streams. This explains why the new protocol has a better performance in GT-ITM as shown in Figure 4.1. It is further observed that, in Figure 4.3, the no-cost percentage of GT-ITM increases rapidly till the request rate at around 750 and then slows down, meanwhile the network bandwidth cost of GT-ITM in Figure 4.1(a) reaches the peak and starts to decrease at around the same request rate.

Besides the network bandwidth cost, the server bandwidth cost is studied as well for both the peer-to-peer stream merging protocol and the conventional single-server based HMSM. Figure 4.4 provides the results in both GT-ITM and Inet topologies, which were obtained from the same experiments as above. With respect to the conventional HMSM, the results show that the performance is nearly identical in GT-ITM and Inet, and is consistent with the result obtained in previous study [18] which shows the server bandwidth cost increases logarithmically with the file request rate.

It is also shown in Figure 4.4 that the new protocol yields better performance



(a) GT-ITM (500 network nodes)



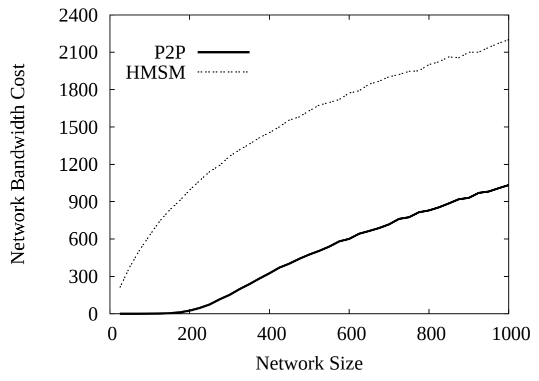
(b) Inet (4000 network nodes)

Figure 4.4: Effect of File Request Rate on Server Bandwidth Cost: Peer-to-peer Stream Merging vs. HMSM (unlimited peer resource capability)

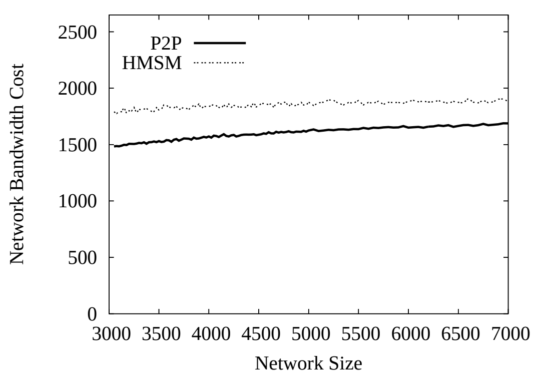
no matter in which type of topology, substantially improving on the conventional HMSM. However, similar to the performance on network bandwidth cost, the performance variance is presented again on server bandwidth cost between GT-ITM and Inet topologies. The protocol yields much better performance in GT-ITM. As the file request rate grows, the server bandwidth cost slightly decreases to 0 in GT-ITM, but slowly increases from 4 to 8 in Inet. A possible reason is an implementation issue related to the difference in internal topology structure between GT-ITM and Inet. In the simulator, the new protocol is implemented in which the server is set by default to be the initial candidate to deliver a stream for a new arrival. The actual initiator will be resulted by comparing all the other candidates with the server based on a particular peer selection policy, which is the *location-first* policy in this experiment. As implied in Figure 4.2(b) that most of the peer candidates are with distance 3 or 4 to the new peer in Inet, there is a higher possibility in Inet for the server to be the actual initiator for the new arrival, thus resulting in more server bandwidth consumed in Inet than in GT-ITM.

4.3.1.2 Effect of Network Size

The next set of experiments studies how protocol performance scales with increase in network size. Figure 4.5 and Figure 4.8 respectively present the results in terms of the network and server bandwidth costs assessed as a function of network size, and give a comparable view of the new protocol with the conventional HMSM.



(a) GT-ITM

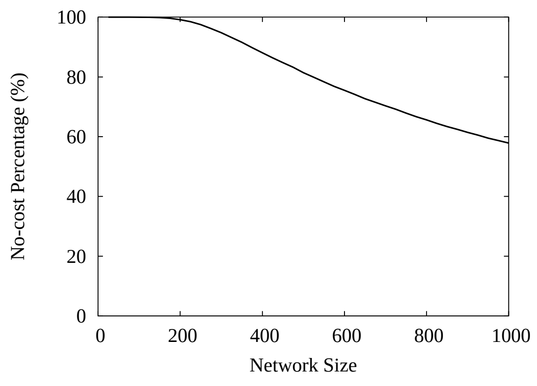


(b) Inet

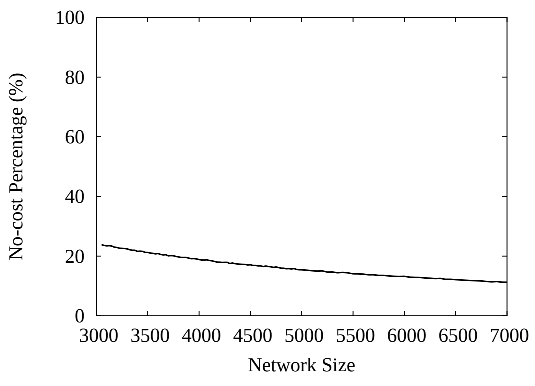
Figure 4.5: Effect of Network Size on Network Bandwidth Cost: Peer-to-peer Stream Merging vs. HMSM (unlimited peer resource capability, file request rate = 1000)

Figure 4.5 presents the network bandwidth costs for the new protocol and HMSM, with the results in GT-ITM and Inet topologies shown in Figure 4.5(a) and Figure 4.5(b), respectively. Some observations are summarized and analyzed as follows:

- The peer-to-peer stream merging protocol yields better performance in both GT-ITM and Inet as the network size increases. Also, similar to the performance on the effect of file request rate (shown in Figure 4.1), better performance is achieved in GT-ITM. Results in Figure 4.5(a) show that the new protocol applied in GT-ITM can save on average over 50% of the network bandwidth consumed in the conventional HMSM, while only about 20% of network bandwidth cost can be saved in Inet as shown in Figure 4.5(b). The reason for this is similar to that analyzed in Section 4.3.1.1 for the performance variance presented on the effect of file request rate. Here, with a fixed file request rate, it is more likely related to the no-cost percentages in network topologies of different sizes. As shown in Figure 4.6, with increase in network size, the no-cost percentage in GT-ITM remains in overall at a much higher level (over 60%) compared to that in Inet (around 20%), which can probably explain the performance variance presented in Figure 4.5.



(a) GT-ITM

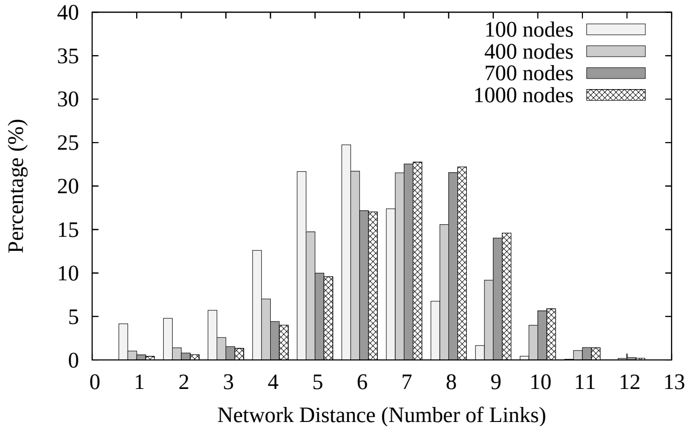


(b) Inet

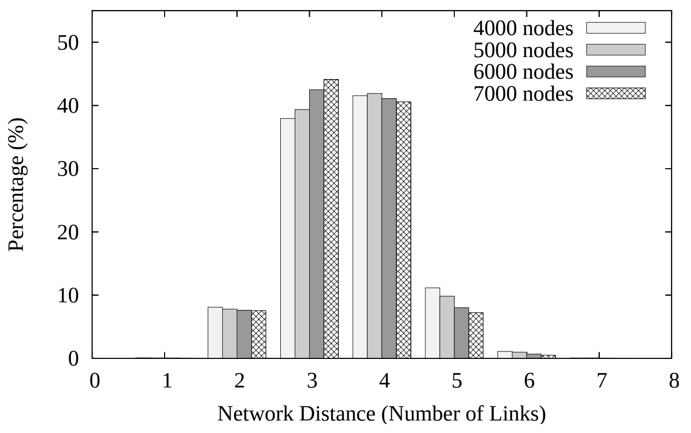
Figure 4.6: Effect of Network Size on No-cost Percentage for Peer-to-peer Stream Merging

- Also observe that, as the network size increases, the percentage of network bandwidth cost saved by the new protocol shows a decreasing trend in both GT-ITM and Inet, with a high decreasing rate in GT-ITM and only a slight decreasing in Inet. As shown in Figure 4.5, the network bandwidth saved in GT-ITM decreases from over 90% to 50% as the network size increases from 200 to 1000, while there is only a minor decrease overall in Inet. Similarly, this is closely related to the no-cost percentages in GT-ITM and Inet, which is illustrated in Figure 4.6. In the figure, the no-cost percentage decreases in both GT-ITM and Inet as the network size increases. Also, compared to GT-ITM in which the no-cost percentage steadily decreases from 100% to 60%, the no-cost percentage in Inet shows a nearly flat decreasing from 22% to 12%. This explains why the network bandwidth saving constantly decreases as the network size increases, and why it is with a different view in GT-ITM and Inet. It can be expected that the network bandwidth saving in GT-ITM would have a similar view to that in Inet when the network size of GT-ITM increases to thousands.

- A further observation from Figure 4.5 is that, with increase in network size,



(a) GT-ITM



(b) Inet

Figure 4.7: Comparing Network Distance Allocation in Topologies of Different Sizes

the network bandwidth cost for both the new protocol and HMSM shows an increasing trend in GT-ITM, but remains almost at the same level in Inet. To explore the reason for this, the network distance allocations in network topologies of four different sizes (100, 400, 700, 1000 nodes for GT-ITM, and 4000, 5000, 6000, 7000 nodes for Inet) are plotted and compared in Figure 4.7 which shows the similar information to that in Figure 4.2. In the figure, the percentages in topologies of four network sizes are clustered at each particular network distance so that the comparison can be easily illustrated. From Figure 4.7(b), observe that, no matter in which size of network topology of Inet, the allocated percentage of node pairs is nearly the same at each distance, which can clearly explain why the network bandwidth cost remains almost unchanged in Figure 4.5. However, this is not the case for GT-ITM topologies. As shown in figure 4.7(a), the percentage of node pairs in four sizes of topologies, especially the 100-node and 400-node topologies, is obviously different at each distance, with higher percentages allocated at smaller distances for smaller-size topologies (e.g., higher percentages at distance 5 and 6 for the 100-node topology), and at larger distances for larger-size topologies (e.g., higher percentages at distance 7 and 8 for the 700-node topology). Based on that, as the network size increases, it is more likely for nodes to have a longer distance between each other, and thus costs more network bandwidth for stream delivery, as shown in Figure 4.5.

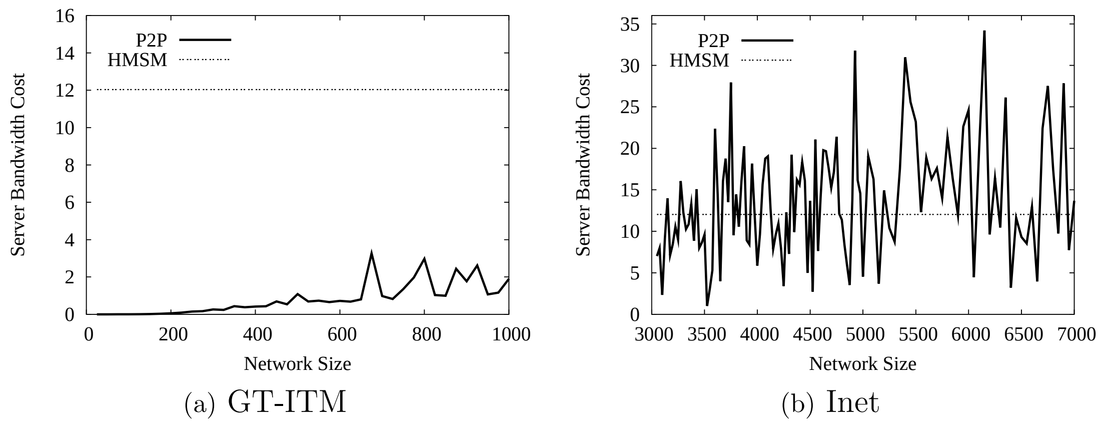


Figure 4.8: Effect of Network Size on Server Bandwidth Cost: Peer-to-peer Stream Merging vs. HMSM (unlimited peer resource capability, file request rate = 1000)

For the server bandwidth cost, Figure 4.8 presents the results in both GT-ITM and Inet topologies. The results show that the server bandwidth for HMSM is constant and of exactly same value in both GT-ITM and Inet. This can be expected since the server bandwidth cost is irrelevant to the network size and the type of topology, and only changes with the file request rate as shown in the previous study

[18]. Since the file request rate is fixed at 1000 in experiments for both GT-ITM and Inet, the obtained server bandwidth cost is identical in both types of topologies.

Regarding the new protocol, the results in both topologies fluctuate frequently and show no explicit regular pattern, and are more widely ranging in Inet. Compared to HMSM, the new protocol saves much more server bandwidth cost in GT-ITM, and achieves little saving in Inet. The reason for the variance between GT-ITM and Inet is discussed in the analysis of Figure 4.4. Although there is no obvious benefit from using the new protocol in Inet in terms of the server bandwidth cost, the performance on average is at least close to that in the conventional HMSM which is also considered to be efficient.

4.3.2 Comparing Peer Selection Policies

A peer selection policy is a key component in the design of peer-to-peer stream merging. It is essential when selecting stream initiators for new arrivals. As described in Section 3.3.1, the *location-first* policy aims at minimizing the network bandwidth cost, the *load-first* policy considers solely the workload balance on delivering media streams, and the *location-bandwidth ratio* policy attempts to achieve a balance between the network bandwidth cost and the workload. With the experimental results presented, this section studies and compares the performance of these three selection policies, attempting to find the best fit to stream merging applied in a peer-to-peer environment.

For the consistency of stream merging policy, the extended stream merging policy based on ERMT, which is expected to be efficient in terms of the network bandwidth cost, is used in all experiments that were conducted for the comparison of three peer selection policies. In all experiments, each peer is assumed to be with unlimited outbound bandwidth and unlimited buffer space. Also, instead of the available outbound bandwidth defined in the *location-first* policy and the workload of a peer defined in the *load-first* policy, the actual number of streams a peer is delivering is measured in the experiments. Since the outbound bandwidth of a peer is defined as units of the stream delivery bit rate as described in Section 4.2.3.2, and the network distance between two peers is measured as the number of links, there is a possibility that more than one peer could be generated by a peer selection policy. For example, a peer with location-bandwidth ratio of 2/4 (i.e., distance 2 and bandwidth 4), and another peer with a ratio of 1/2, will be of same interest to the *location-bandwidth ratio* policy. In such cases, it is assumed for both the *location-bandwidth ratio* and the *load-first* policies that, if more than one peer resulted in a policy, the one who is located at the same node as the requesting peer (i.e., with distance 0) will be selected; otherwise if there is no such peer, one will be selected at random from all resulted peers.

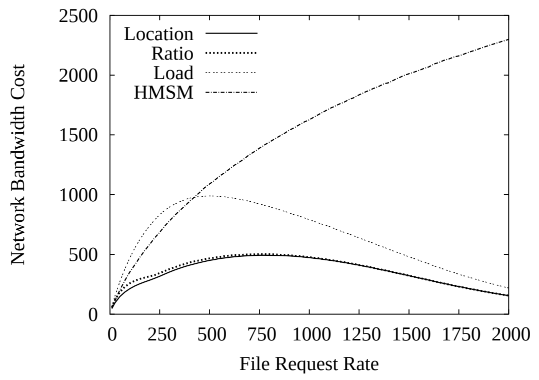
With the main focus on the delivery efficiency, Section 4.3.2.1 and Section 4.3.2.2 study the effects of file request rate and network size, concerning mostly on performance in terms of the network and server bandwidth costs. The issue of workload balancing is studied in Section 4.3.2.3 with the illustration of corresponding performance in three peer selection policies. In all figures presented in this section, if it is not further specified, the lines labelled by "Location" represent results for the *location-first* policy, "Ratio" is for the *location-bandwidth ratio* policy, and "Load" is for the *load-first* policy.

4.3.2.1 Effect of File Request Rate

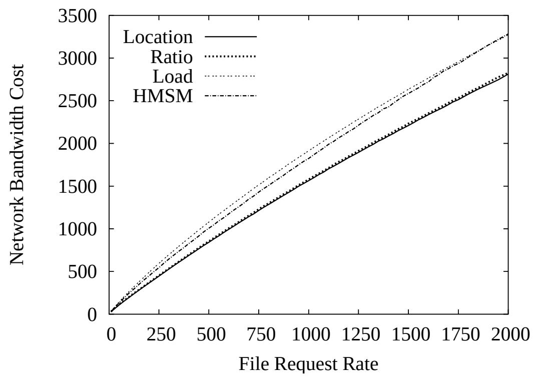
In this set of experiments, the performance of three peer selection policies are evaluated in terms of the effect of file request rate. With results illustrated in Figure 4.9, Figure 4.11, and Figure 4.12, these policies are compared in terms of various bandwidth costs assessed as a function of file request rate.

Figure 4.9 provides the network bandwidth costs for the three peer selection policies in both GT-ITM and Inet topologies. Corresponding results of single-server based HMSM is also presented in the figure for comparison. Some observations obtained from the figure are summarized as follows:

- Regarding the performance on network bandwidth cost, three peer selection



(a) GT-ITM (500 network nodes)



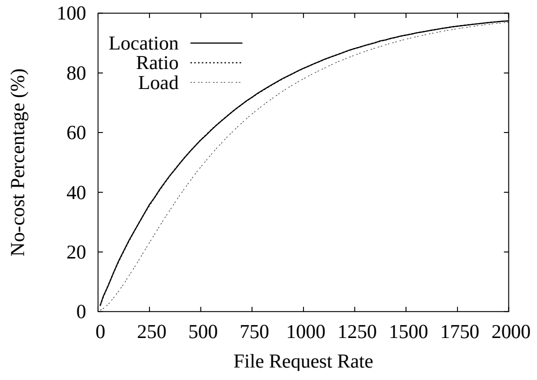
(b) Inet (4000 network nodes)

Figure 4.9: Effect of File Request Rate on Network Bandwidth Cost: Peer Selection Policies (unlimited peer resource capability)

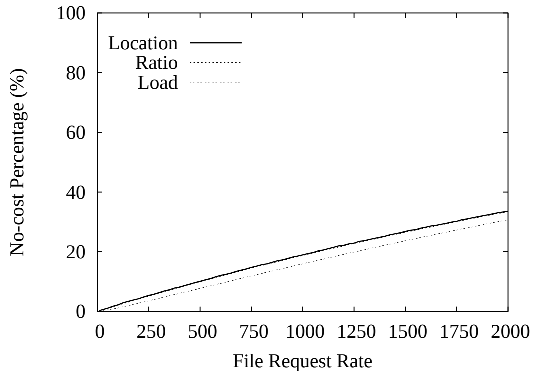
policies yield similar scaling as the file request rate increases. In GT-ITM, as shown in Figure 4.9(a), the network bandwidth costs of three policies all increase at lower request rates and then gradually decrease after they reach the corresponding peaks; and in Inet, they similarly remain an increasing trend within the range of file request rate used in the experiment, as shown in Figure 4.9(b). The reasons for the performance variance between GT-ITM and Inet are discussed in Section 4.3.1.1.

- Comparing three selection policies, no matter in which type of topology, 1) the protocol with the *location-first* policy and that with the *location-bandwidth ratio* policy achieve almost the same performance; and 2) the *load-first* policy is the least efficient when the protocol is evaluated in terms of the network bandwidth cost, which can be expected from the policy definition described in Section 3.3.1.

- The new protocol with either the *location-first* or the *location-bandwidth ratio* policy consumes less network bandwidth than the conventional HMSM protocol in both GT-ITM and Inet. However, the new protocol with the *load-first* policy performs differently. Compared with the conventional HMSM protocol, it does not always consumes less network bandwidth within the range of request rate measured in GT-ITM, and almost always consumes more network bandwidth in Inet.



(a) GT-ITM (500 network nodes)

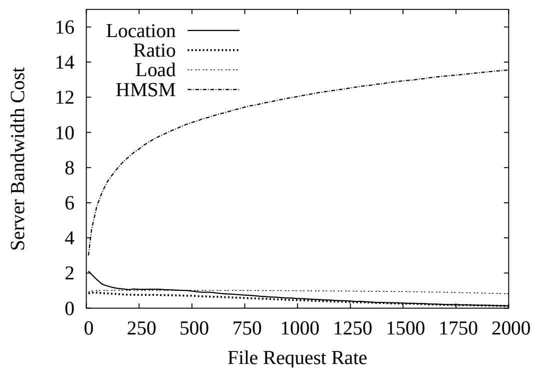


(b) Inet (4000 network nodes)

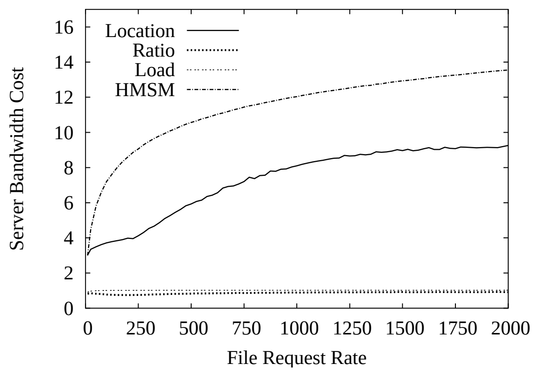
Figure 4.10: No-cost Percentage for Three Peer Selection Policies

To provide a further explanation, no-cost percentages for these three selection policies are also studied, and results are plotted in Figure 4.10. Compared with Figure 4.9, consistency can be observed in Figure 4.10, i.e., the no-cost percentage presented for the *load-first* policy is lower than those for the other two policies, resulting in more network bandwidth cost consumed in protocol with the *load-first* policy, as illustrated in Figure 4.9. Moreover, nearly fully overlapped no-cost percentages are observed for the *location-first* and *location-bandwidth ratio* policies, which is certainly related to the same performance of two policies as shown in Figure 4.9. The *location-first* policy is expected to be the most efficient in terms of the network bandwidth cost among the three policies. However, under the assumption of unlimited peer outbound bandwidth, there is a big possibility for a new arrival to find another peer who is not only nearby but also with less workload, to create and deliver a stream, and thus the *location-bandwidth ratio* policy yields the same performance as the *location-first* policy.

The server bandwidth costs for the three peer selection policies and the conventional HMSM are provided in Figure 4.11, including results in both GT-ITM and Inet topologies. Overall, the new protocol with either of the three selection policies



(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)

Figure 4.11: Effect of File Request Rate on Server Bandwidth Cost: Peer Selection Policies (unlimited peer resource capability)

performs better than the conventional HMSM protocol. This is expected, since the task of delivering streams is distributed among all participating peers in the new protocol. However, distinct performance can be identified among the three policies, as discussed below:

- The protocol with the *location-bandwidth ratio* policy performs similarly to that with the *load-first* policy in both GT-ITM and Inet, i.e., the consumed server bandwidths of both policies are much less than that consumed in conventional HMSM. Since both policies partially or completely concern about the workload balance among all participating peers, it is reasonable for the server, which can also be considered as one of the participating peers, to consume bandwidth on average less than 1. This implies that, with good workload balance, the capability of delivering one media stream for each active peer is sufficient for the overall system to achieve good performance. As with the *location-first* policy, the protocol performs differently in GT-ITM and Inet. The reason for this performance variance is explained in the analysis for Figure 4.4 in Section 4.3.1.1, which is mainly due to the difference in topology structure. It exposes the weakness of a location-only policy, i.e., the randomness regarding the number of streams that a peer is asked to deliver. This number could be unacceptably large for a normal peer in some type of topology, such as Inet, in which most of the distances between two nodes are similar.

- Regarding the location issue, it has a positive effect on the server bandwidth cost. As shown in the figure, although the protocol with *load-first* policy performs well with the server bandwidth cost close to 1, the *location-bandwidth ratio* policy is still a bit more efficient in both GT-ITM and Inet. Also, in a topology in which the distances between two nodes are dispersedly distributed (e.g., GT-ITM), the server bandwidth costs for the two location-related policies (i.e., *location-first* and *location-bandwidth ratio*) all show a decreasing trend as the file request rate grows.

As one of the important measures for evaluating peers' contribution to the media delivery, the total peer bandwidth cost is also studied in this experiment. Figure 4.12 provides the peer bandwidth costs for the three peer selection policies, and illustrates how these costs scale with increase in file request rate. Three major observations are described as follows:

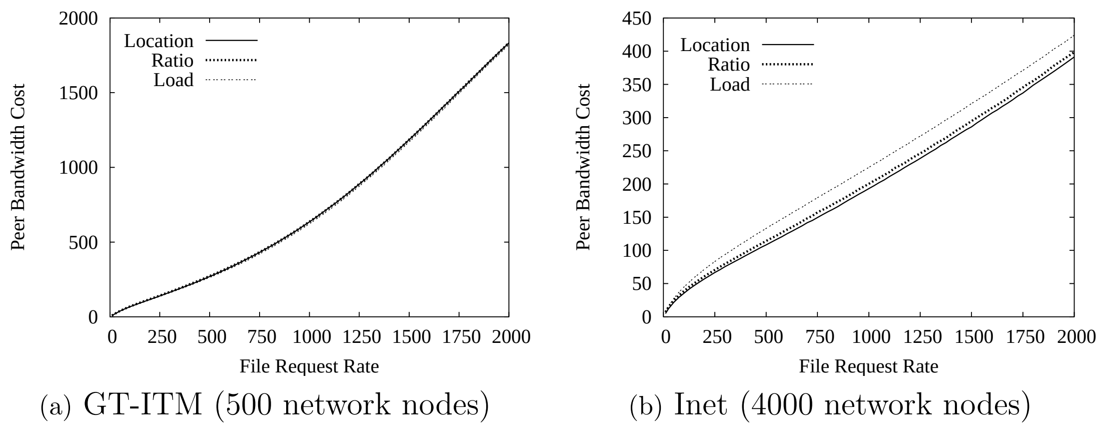
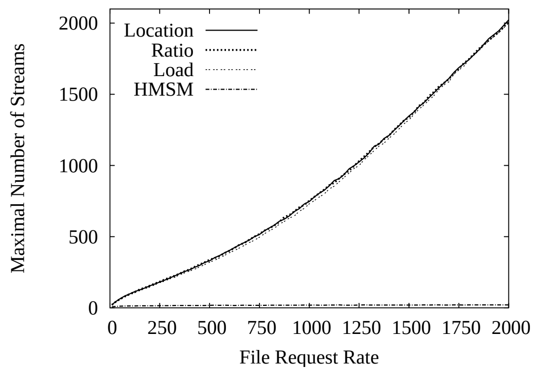
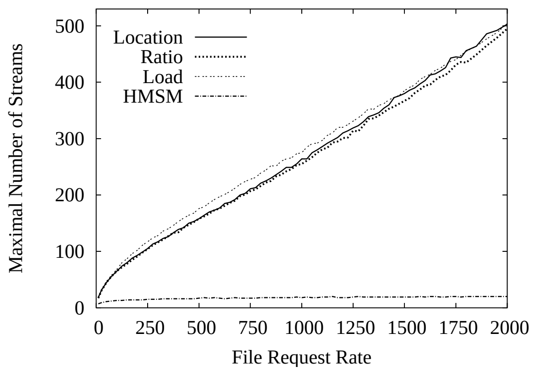


Figure 4.12: Effect of File Request Rate on Peer Bandwidth Cost: Peer Selection Policies (unlimited peer resource capability)

- In both GT-ITM and Inet, the peer bandwidth costs for all the three selection policies steadily increase as the file request rate grows. As recalled, the total number of active streams in a single-server based HMSM system remains low since streams constantly merge with previous streams. However, it is not the same case in the new protocol. As described in Section 3.3.2.3, a peer who is listening on a no-cost stream will listen to only that stream till the stream reaches the end of the media content. It implies that the total number of active streams in a peer-to-peer stream merging system is much more than that in a conventional HMSM system, and highly depends on the percentage of no-cost streams. Now that almost all the streams are delivered by peers in the new protocol, the total peer bandwidth consumed on delivering streams is closely related to the total number of streams in the system. Figure 4.13 plots the maximums of the total number of active streams for new protocol with the three selection policies and for the conventional HMSM. Observe that, the maximal numbers of streams for protocol with the three policies show the same trend as that for the peer bandwidth costs shown in Figure 4.12, i.e., more streams exist in the system as the file request rate increases, resulting in more peer bandwidth consumed in media delivery. Also observe that, the maximal number of streams in the conventional HMSM is almost constant and remains low, which is consistent with the expectation.



(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)

Figure 4.13: Maximal Number of Streams for Peer Selection Policies (unlimited peer resource capability)

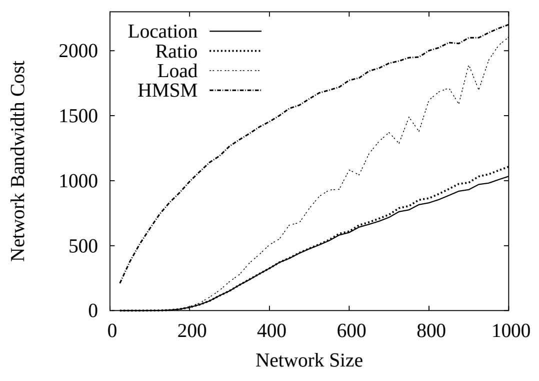
- With either of the three policies, the new protocol consumes much less peer bandwidth in Inet than in GT-ITM. For example, an approximate peer bandwidth of 1800 is consumed in GT-ITM at file request rate 2000, but only about 400 is consumed in Inet. Since it is mentioned in the previous observation that the peer bandwidth cost highly depends on the percentage of no-cost streams, the performance variance in the peer bandwidth cost is well explained by the difference of no-cost percentage between GT-ITM and Inet (shown in Figure 4.10). With a higher percentage of no-cost streams in GT-ITM, more streams don't merge with any other streams, i.e., more active streams exist in the system as shown in Figure 4.13, and thus more peer bandwidth is consumed.

- Comparing three selection policies, there is no obvious difference in the peer bandwidth cost. As shown in Figure 4.12(a), the corresponding peer bandwidth costs are almost completely overlapped in GT-ITM. Although Figure 4.12(b) shows that the protocol with the *load-first* policy consumes comparatively the most peer bandwidth in Inet, followed in descending order by the *location-bandwidth ratio* policy and the *location-first* policy, the differences among the three policies are minor. Considering the difference in network size, which causes the difference in no-cost percentage between two types of topologies, one can expect that the peer bandwidth cost in Inet will show a similar view to that in GT-ITM if the file request rate is sufficiently high.

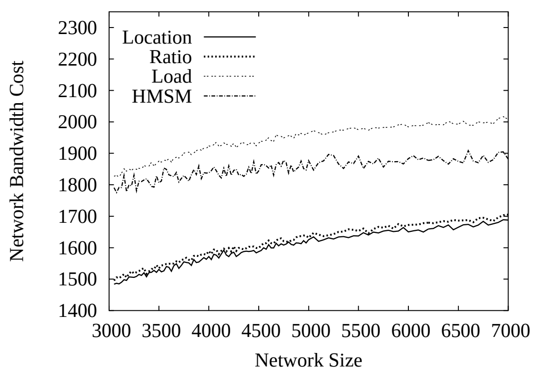
4.3.2.2 Effect of Network Size

The set of experiments presented in this section compares the performance of three peer selection policies in terms of the effect of network size. Various bandwidth costs are studied and presented in Figure 4.14, 4.17, and 4.18, assessed as a function of network size.

Figure 4.14 provides the network bandwidth costs for the new protocol with three peer selection policies and for the conventional HMSM, including results in both GT-ITM and Inet topologies. The following summarizes some observations obtained from the figure:



(a) GT-ITM

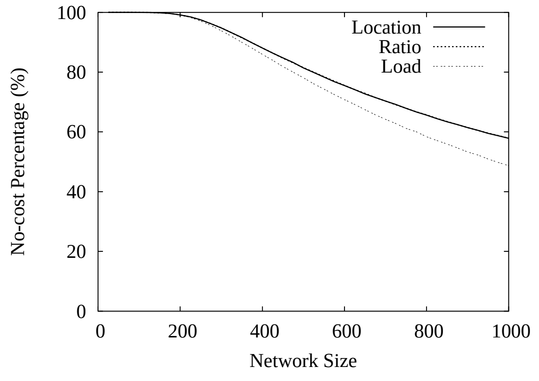


(b) Inet

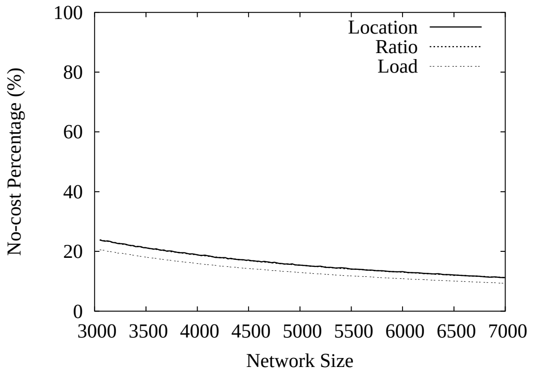
Figure 4.14: Effect of Network Size on Network Bandwidth Cost: Peer Selection Policies (unlimited peer resource capability, file request rate = 1000)

- Similar to the results in terms of the file request rate, the network bandwidth cost consumed in the protocol with the *location-bandwidth ratio* policy is nearly identical to that consumed with the *location-first* policy, which is only slightly more in Inet and in large network size of GT-ITM topology. Compared with the conventional HMSM, the new protocol with these two location-related policies saves considerable amount of network bandwidth cost, which implies that the location is the dominant factor for the new protocol to achieve good performance in terms of the network bandwidth cost. More analysis on the comparison with HMSM is presented in Section 4.3.1.2.

- The *load-first* policy is the least efficient among the three policies. As Figure 4.14(a) shows, in GT-ITM, the network bandwidth consumed with this policy shows a faster increasing trend compared with the other two policies. Although it is still overall less than that consumed in the conventional HMSM in GT-ITM, it will not likely remain so if a wider range of network size is measured. Moreover, as shown in Figure 4.14(b), the overall network bandwidth consumed in Inet is even more than that consumed in the conventional HMSM. It appears that there should be no cause for the new protocol with either of

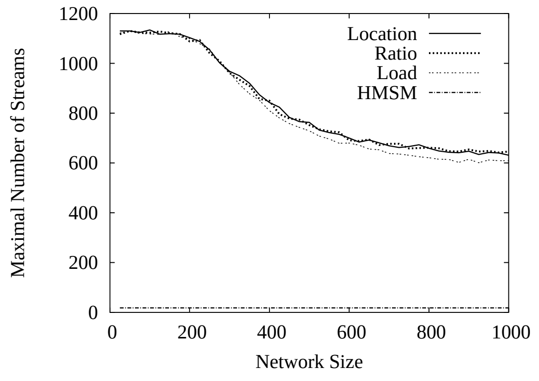


(a) GT-ITM

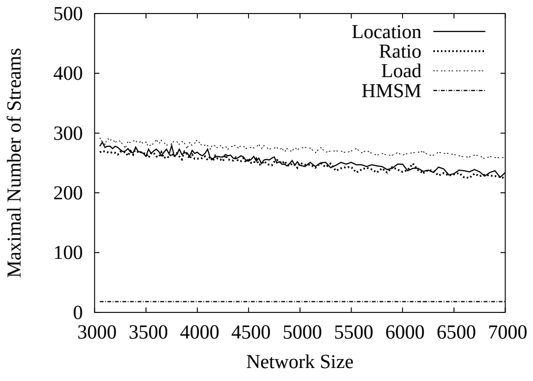


(b) Inet

Figure 4.15: No-cost Percentage for Three Peer Selection Policies



(a) GT-ITM



(b) Inet

Figure 4.16: Maximal Number of Streams for Peer Selection Policies (unlimited peer resource capability, file request rate = 1000)

the three policies to perform worse than the conventional HMSM in the first place, since the high percentage of no-cost streams in the new protocol (shown in Figure 4.15) results in the network bandwidth savings from the conventional HMSM in which there is almost no no-cost stream. However, from another point of view on the total number of streams in the system, there surely exists some inefficiency in the new protocol, especially with the *load-first* policy. As shown in Figure 4.16, the maximal number of streams in the new protocol is much more than that in the conventional HMSM. When the new protocol applies the *load-first* policy, a considerable number of streams are delivered by peers who are far from the requesting peers, causing substantial redundant use of network links, compared with the conventional HMSM in which the number of streams remains low and each stream is delivered via efficient multicast.

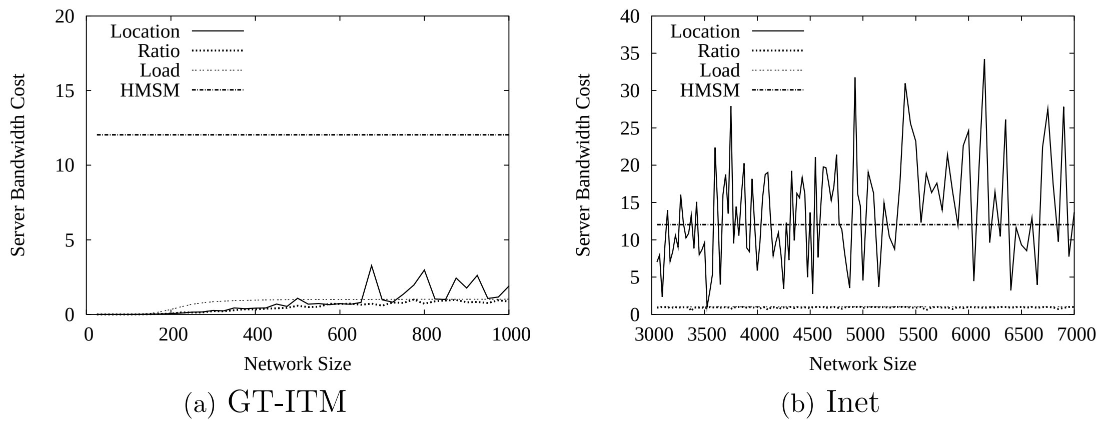
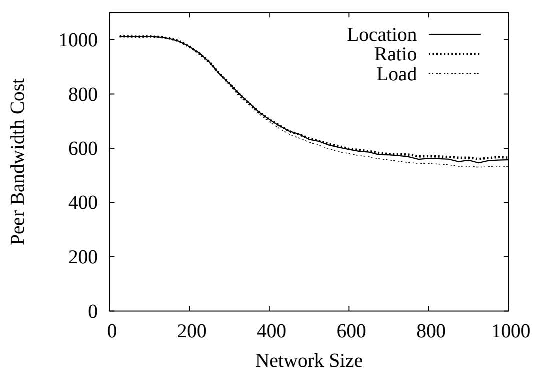


Figure 4.17: Effect of Network Size on Server Bandwidth Cost: Peer Selection Policies (unlimited peer resource capability, file request rate = 1000)

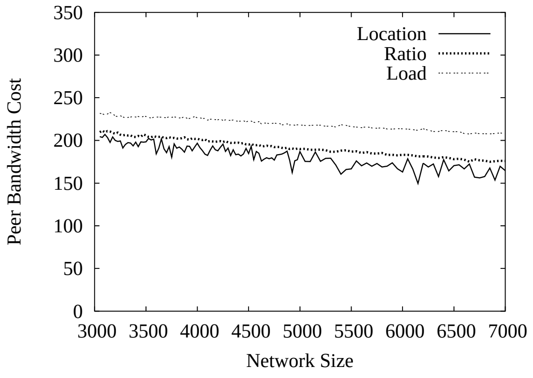
Figure 4.17 provides the server bandwidth cost for the three selection policies with increase in network size. Observe that, compared to the server bandwidth costs for the *location-bandwidth ratio* policy and the *load-first* policy which are stable and around 1, the cost for the *location-first* policy is higher and fluctuates frequently in Inet and in large network size of GT-ITM. Such fluctuation implicitly exposes the workload imbalance in the protocol with the *location-first* policy. Also observe that, the protocol with both load-related (*location-bandwidth ratio* and *load-first*) policies consumes much less server bandwidth cost than the conventional HMSM, which demonstrates the benefit of using workload balance in a peer selection policy. For the performance variance of the *location-first* policy in GT-ITM and Inet, the reader is referred to the analyses of Figure 4.4 and Figure 4.8 for more details.

Figure 4.18 provides the peer bandwidth costs for the three selection policies. Two major observations are described as follows:

- With a fixed file request rate of 1000, it is shown that the peer bandwidth cost is extremely high in GT-ITM when the network size is less than 200. This is



(a) GT-ITM



(b) Inet

Figure 4.18: Effect of Network Size on Peer Bandwidth Cost: Peer Selection Policies (unlimited peer resource capability, file request rate = 1000)

expected, since the request rate of 1000 is sufficiently high for such a small network size, almost every peer can obtain a no-cost stream from another peer who is located at the same node, resulting in a extremely high percentage of no-cost streams as shown in Figure 4.15, and thus in high peer bandwidth cost. When the network size continuously grows after 200, the peer bandwidth cost gradually declines and is stabilized approximately at 600. For Inet, since the starting network size measured is 3050, which is considerably large for the fixed request rate of 1000, the possibility is low for a peer to obtain a no-cost stream. Thus, the overall peer bandwidth costs remain relatively stable within the range of the network size measured.

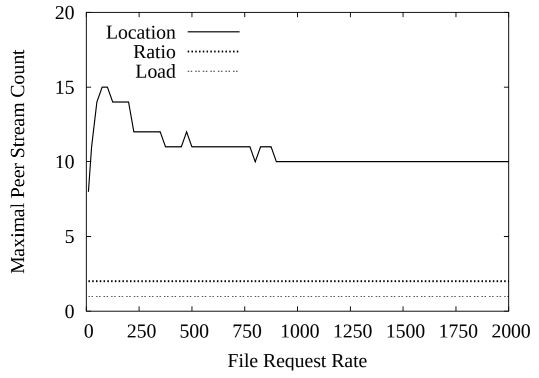
- Comparing the three selection policies, the corresponding three peer bandwidth costs scale very similarly as the network size increases, and are very close in GT-ITM and not significantly different in Inet, as shown in Figure 4.18. Also observe that the protocol with the *load-first* policy performs differently in GT-ITM and Inet. One possible reason would be the difference in internal structure between the two types of topologies. More study is needed in the future work for further investigation on this issue.

4.3.2.3 Workload Balance

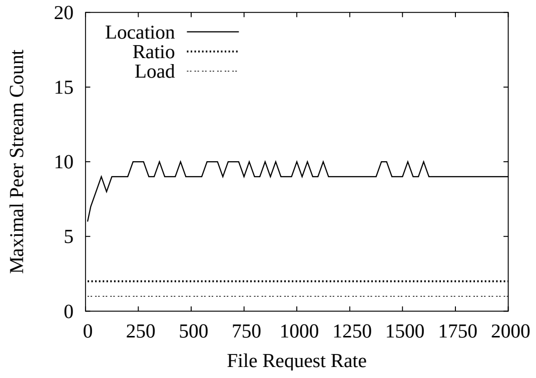
The previous two subsections study the performance of the peer selection policies mainly on the various bandwidth costs. Results show, with the two location-related policies, the new protocol performs similarly well on the network bandwidth cost, but differently on the server bandwidth cost. The *location-bandwidth ratio* policy that concerns both the location and the workload balance, is more efficient in terms of the server bandwidth cost, which implies the workload is also an issue that may affect the performance of the new protocol. To further investigate the workload issue as a supplement to this performance study, this section provides some sample results that illustrate and compare how the workload is distributed among the three peer selection policies.

The workload in the peer-to-peer stream merging system refers to the task of delivering streams. The workload is assigned to all active peers. Specifically, the workload of a particular peer is defined as the number of streams the peer is delivering, and the balance issue of system workload refers to how the total number of streams is distributed among all participating peers. One thing to note is that the number of streams delivered by a peer dynamically changes over time. Rather than the average workload of a peer during its life time, the maximal number of streams that a peer delivers is measured in the experiment, since a extremely high workload on a peer at a certain point of time may affect the overall system workload balance and will likely cause stream disruption. Figure 4.19 provides the maximal peer stream count (i.e., the maximal number of streams delivered by a peer among all participating peers) for the three selection policies, assessed as functions of file request rate and network size.

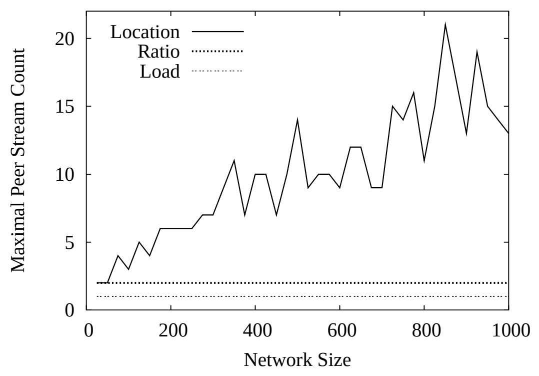
Observe that, in both figures, the maximal peer stream counts for both the load-related policies are constant and remain at low values – 2 for the *location-bandwidth ratio* policy and 1 for the *load-first* policy. It implies that, for each peer, the outbound bandwidth for delivering at most 2 streams is sufficient for the entire system to be effective if the peer workload is well balanced. However, for the *location-first*



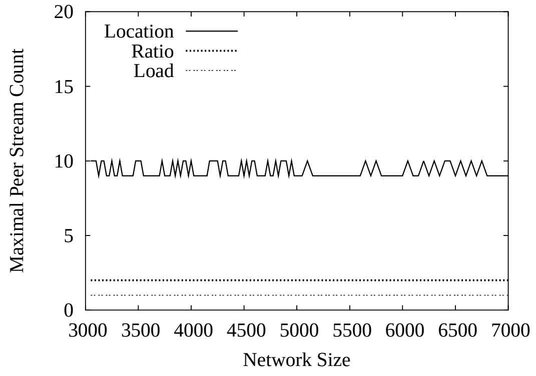
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)



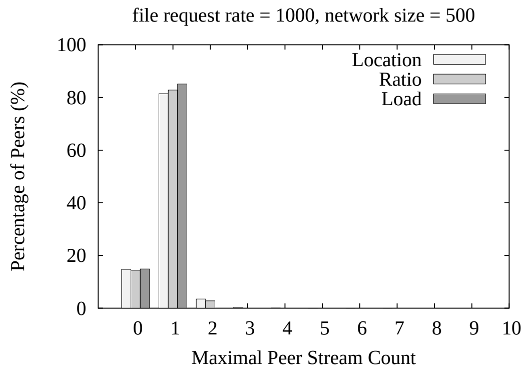
(d) Inet (request rate = 1000)

Figure 4.19: Maximal Peer Stream Count for Peer Selection Policies

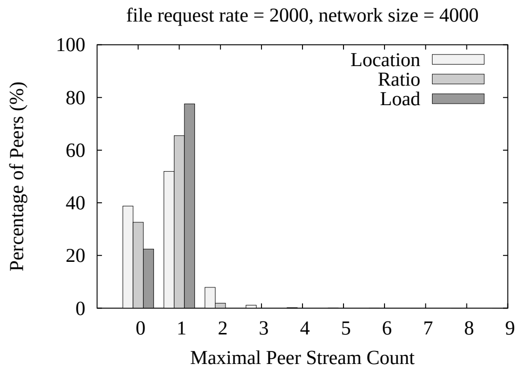
policy, although it is the most efficient in terms of the network bandwidth cost, its maximal peer stream count is unacceptably high in both GT-ITM and Inet, which is impractical in a real system.

The experiment collected the maximal number of streams that each peer delivers. For all active peers, there is an allocation of percentage of peers in terms of the maximal number of streams. As a further reference, Figure 4.20 provides two sample results of this allocation for the three policies. The maximal peer stream count for the *location-first* policy is used as the maximal axis value. Two observations are described as follows:

- Although the maximal peer stream count for the *location first* policy is 9 or 10 as shown in the figure, almost all peers are with maximal stream count less



(a) GT-ITM



(b) Inet

Figure 4.20: Percentage of Peers in Stream Count Allocation for Peer Selection Policies

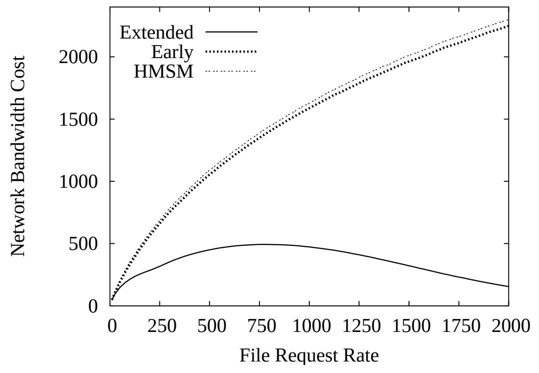
than 3. For the percentages of peers at maximal stream counts larger than 2, the results are too small to be visible.

- No matter which policy is used, most of the peers are with maximal stream count of 1, and there is a considerable number of peers who don't participate in delivering streams, which implies the abundance of resource for contributing to stream delivery.

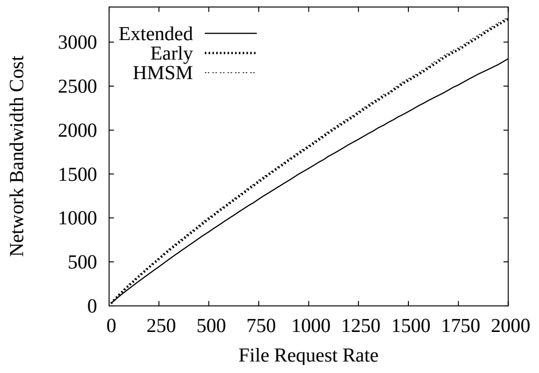
4.3.3 Comparing Stream Merging Policies

A stream merging policy is essential for any stream merging protocol to be effective and efficient. Conventionally, it follows the heuristic of *early merging* to achieve soonest stream mergers which can greatly reduce the required server bandwidth. In peer-to-peer stream merging, the conventional early merging policy is extended to be adaptable to the peer-to-peer context, mainly attempting to improve the efficiency in terms of the network bandwidth cost.

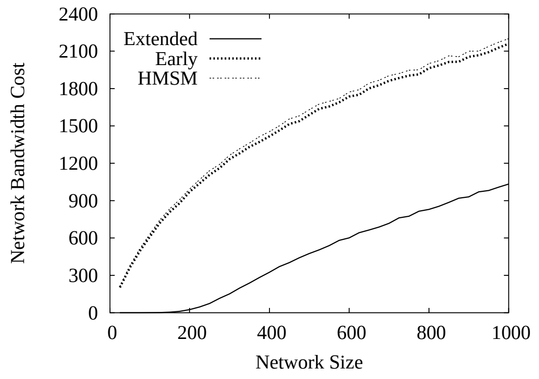
Previously in Section 4.3.1, the new protocol with the extended stream merging policy is compared with the single-server based HMSM that applies one of the conventional early merging policies (i.e., ERMT). In this section, further comparison is presented with the conventional ERMT and the extend stream merging policy respectively applied in the new protocol and combined with the *location-first* policy. Again, each peer is assumed to be with unlimited outbound bandwidth and unlimited buffer space. In all figures presented in this section, unless stated otherwise, the lines labelled by "Extended" represent results for the new protocol with the extended stream merging policy and those labelled by "Early" are for the new protocol with the conventional early merging policy (ERMT).



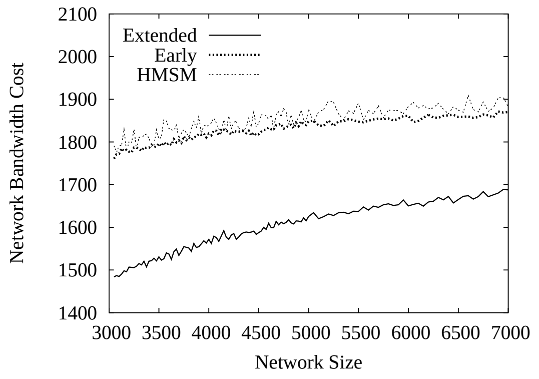
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)

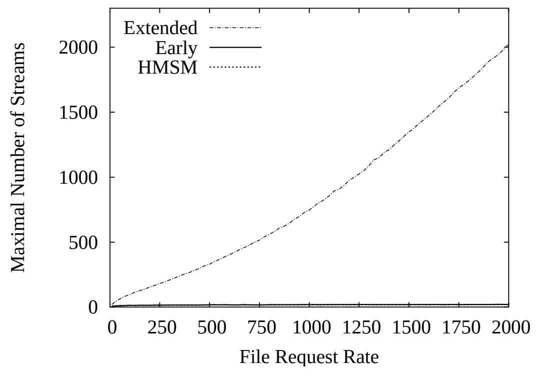


(d) Inet (request rate = 1000)

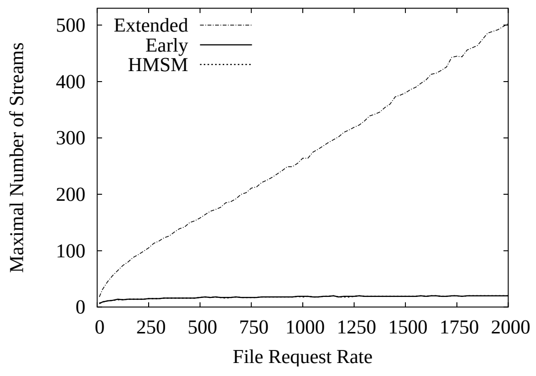
Figure 4.21: Network Bandwidth Cost: Stream Merging Policies (unlimited peer resource capability)

Figure 4.21 provides the network bandwidth costs for both types of stream merging policies, assessed as functions of file request rate and network size. Two observations are described as follows:

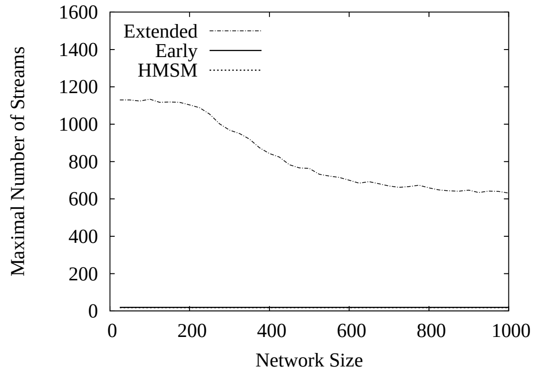
- In terms of the network bandwidth cost, the extended stream merging policy is more efficient than the conventional early merging policy when both are applied in the new protocol. As defined in the conventional early merging policy, each peer, without any exception, has to listen to a merge target stream no matter whether or not the merge target is delivered by a remote peer. Comparatively, more network bandwidth can be saved in the extended stream merging policy, since the merge target is delivered by a close-by peer if a peer successfully obtains one, otherwise, the peer listens to its primary stream only.



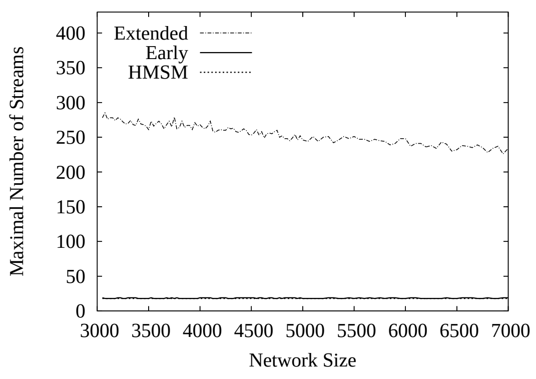
(a) GT-ITM



(b) Inet



(c) GT-ITM



(d) Inet

Figure 4.22: The Maximal Number of Streams: New Protocol with Early Merging Policy vs. HMSM

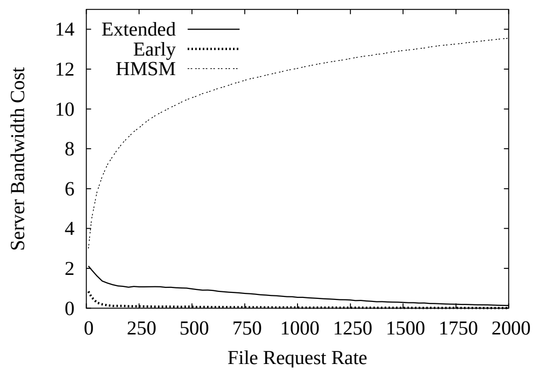
- Also, interestingly, the new protocol with the conventional ERMT consumes the nearly identical network bandwidth to the conventional HMSM, no matter in GT-ITM or in Inet. When the early merging policy is applied, the new protocol has a similar stream merging structure to the conventional HMSM, except that all streams are delivered by peers instead of the server itself only.

Unlike the extended stream merging policy that allows a peer to listen to only its primary stream, it requires each peer to listen to two streams so that streams constantly merge with previous streams, which leads to a low number of active streams in the system. This can be verified by Figure 4.22 that illustrates the maximal numbers of streams for the new protocol with two stream merging policies and the conventional HMSM. It shows that the results for the new protocol with ERMT and the conventional HMSM are fully overlapped, representing the same numbers of streams in both cases. Overall, this implies that the stream merging policy is also a key factor that affects the network bandwidth cost. For the new protocol that applies the early merging policy, although the use of the *location-first* policy saves network bandwidth, the protocol is still not efficient in terms of the total network bandwidth cost.

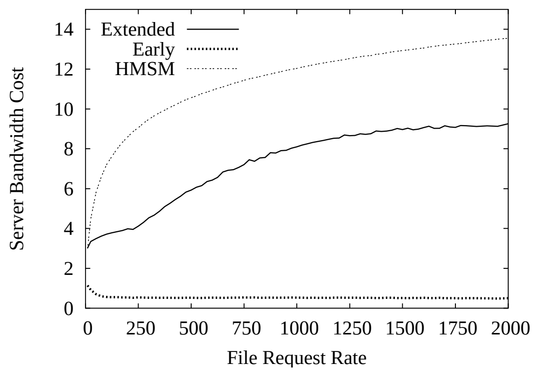
Figure 4.23 and Figure 4.24 illustrate the comparison of both the early and the extended stream merging policies applied in the new protocol in terms of the server/peer bandwidth cost. Comparing two stream merging policies, the main observation is that, the protocol with the early merging policy consumes less server bandwidth cost and much less peer bandwidth cost. As mentioned, the total number of active streams in peer-to-peer stream merging is closely related to the total peer bandwidth cost. It is also related to the server bandwidth cost since the server in the new protocol behaves more like a normal peer than a conventional dedicated server. Now that the total number of streams in the new protocol with the early merging policy is significantly less than that with the extended stream merging policy, as shown in Figure 4.22, it is reasonable for the protocol with the early merging policy to consume less peer/server bandwidth cost.

4.3.4 Impact of Limited Peer Outbound Bandwidth

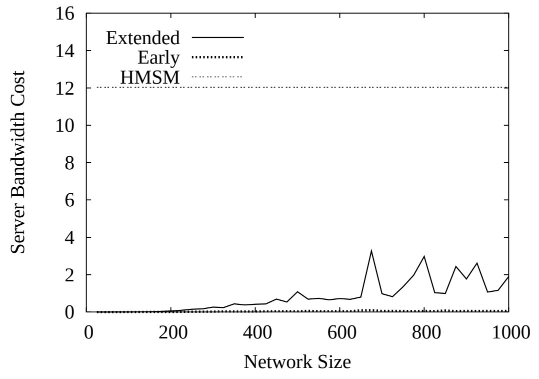
So far, the performance study is based on the assumption of that each peer has unlimited outbound bandwidth that allows a peer to deliver a large number of streams whenever required to optimize the protocol performance. For the peer-



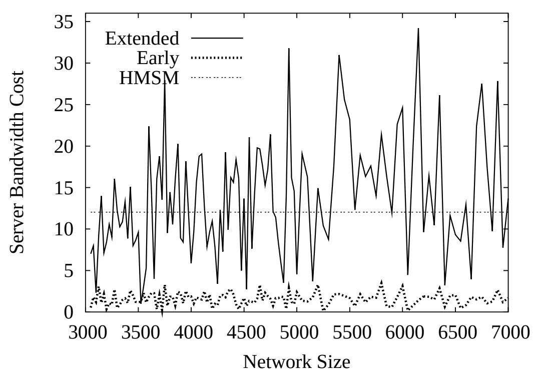
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)

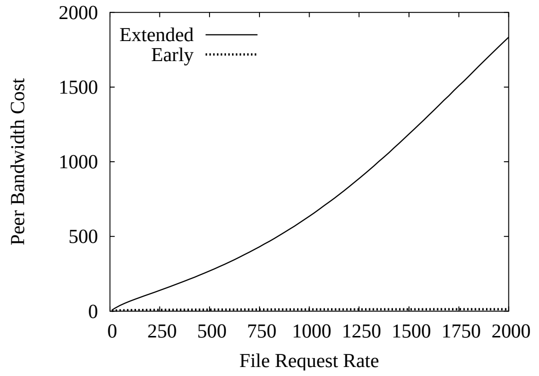


(d) Inet (request rate = 1000)

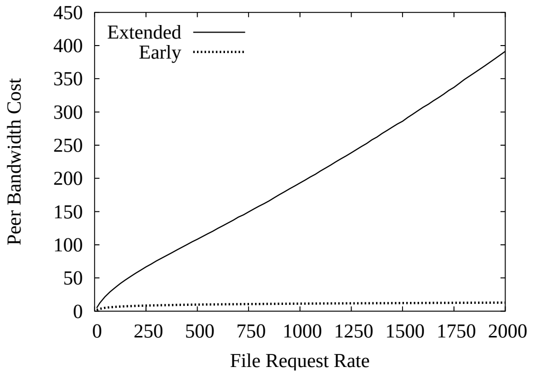
Figure 4.23: Server Bandwidth Cost: Stream Merging Policies

to-peer stream merging technique, more realistically, if a peer generated by the *location-first* policy doesn't have sufficient outbound bandwidth to deliver a stream requested by a new peer, the stream will be simply delivered by a more distant peer, which requires more network bandwidth for the stream delivery. In this section, such impact on the protocol performance is studied.

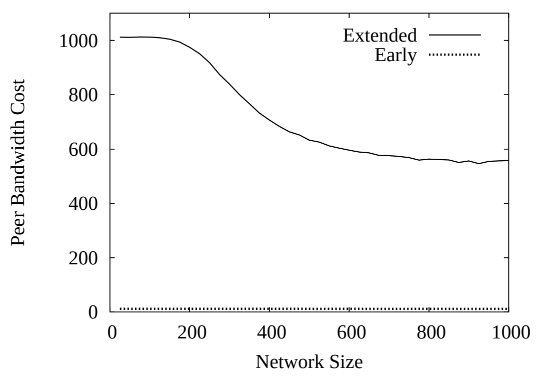
With the default policy and parameter settings, the impact of limited peer outbound bandwidth is studied in both homogeneous and heterogeneous settings. While all peers are simply assigned the same outbound bandwidth capacity in the homogeneous setting, it is more complicated in the heterogeneous setting. Specifically, with an expected value of the limited outbound bandwidth capacity assigned in each simulation run, the system keeps track of the current outbound bandwidth capacity



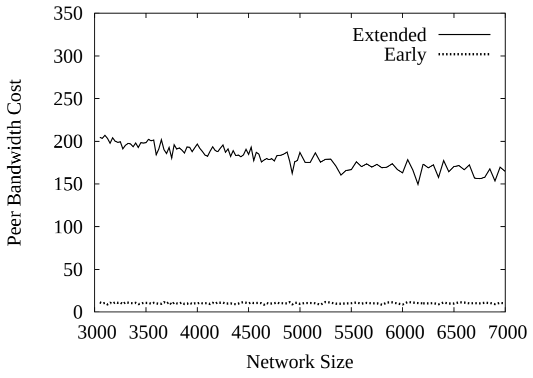
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)



(d) Inet (request rate = 1000)

Figure 4.24: Peer Bandwidth Cost: Stream Merging Policies

which is simply the average of the outbound bandwidth capacities over all active peers. When a new peer joins, its outbound bandwidth capacity is randomly selected from a range determined by comparing the current average with the expected capacity, and the current average is updated whenever a peer joins or leaves the system. In both settings, the media server is treated as a normal peer and assigned the averaged outbound bandwidth capacity. In the circumstances that all the peers and the server are at their full load which causes no outbound bandwidth available in the system to serve any new peer, the server takes the responsibility.

In the figures presented in this section, the results of the new protocol with both homogeneous and heterogeneous limited outbound bandwidth settings are respectively labelled by "P2P homo" and "P2P hete" followed by a number in Figure

4.25 and Figure 4.27 representing the corresponding averaged outbound bandwidth capacity measured as units of the stream delivery bit rate. The lines labelled by "P2P unlimited" represent the results of the new protocol with unlimited outbound bandwidth, and those labelled by "HMSM" are for the conventional HMSM.

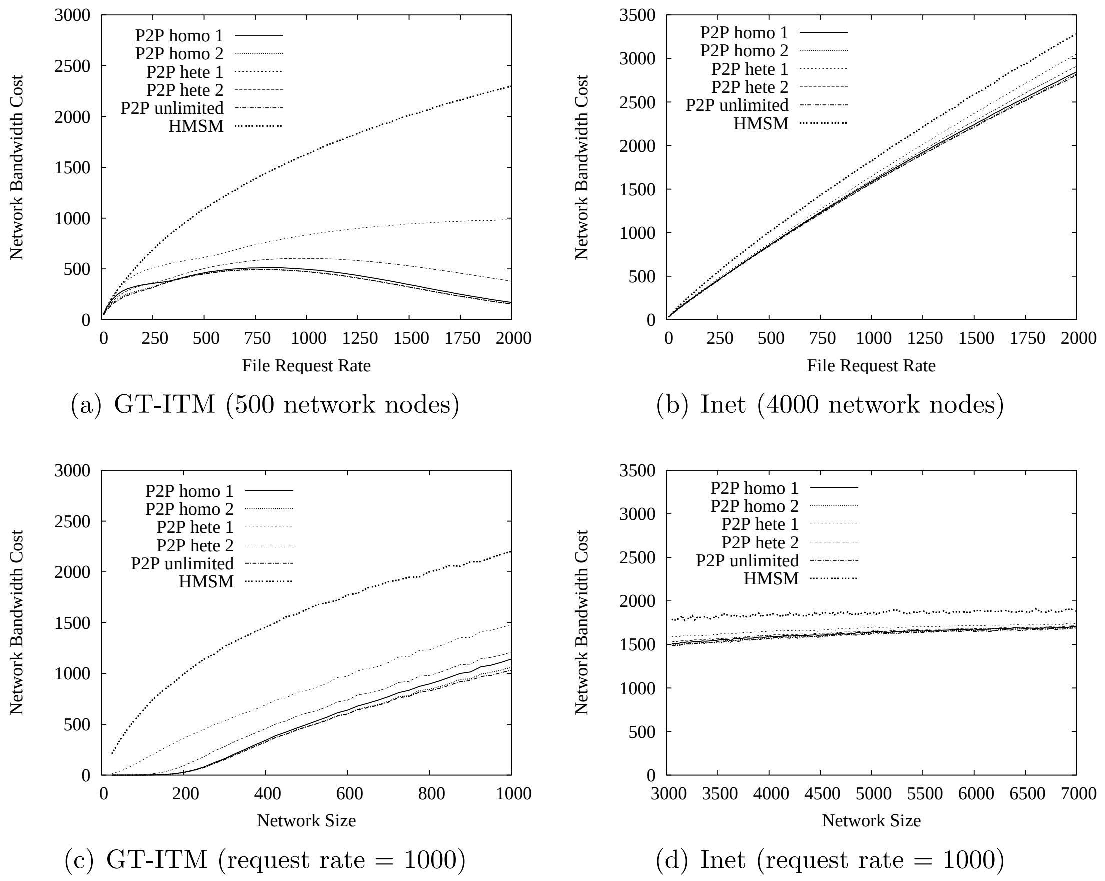


Figure 4.25: Impact of Limited Outbound Bandwidth on Network Bandwidth Cost

Figure 4.25 illustrates the comparison of the new protocol with limited and unlimited outbound bandwidth in both GT-ITM and Inet, in terms of the network bandwidth cost assessed as functions of file request rate and network size. Since it is shown in Figure 4.19 that the capability of delivering one stream for each peer is sufficient for the whole system to be effective, the limited outbound bandwidth capacities of 1 and 2 are respectively used in both the homogeneous and heterogeneous settings. Some observations are described as follows:

- For the outbound bandwidth capacity limited at 1 and 2, the new protocol with the homogeneous setting performs very closely to that with the unlimited outbound bandwidth in terms of the network bandwidth cost, which implies the capability of delivering one stream for a peer is sufficient for the new protocol with homogeneous limited outbound bandwidth to be efficient. In fact, with the homogeneous outbound bandwidth limited at 1, the new protocol based on the *location-first* policy is very similar to the new protocol with the *location-bandwidth ratio* policy that achieves the full balance of the system workload.

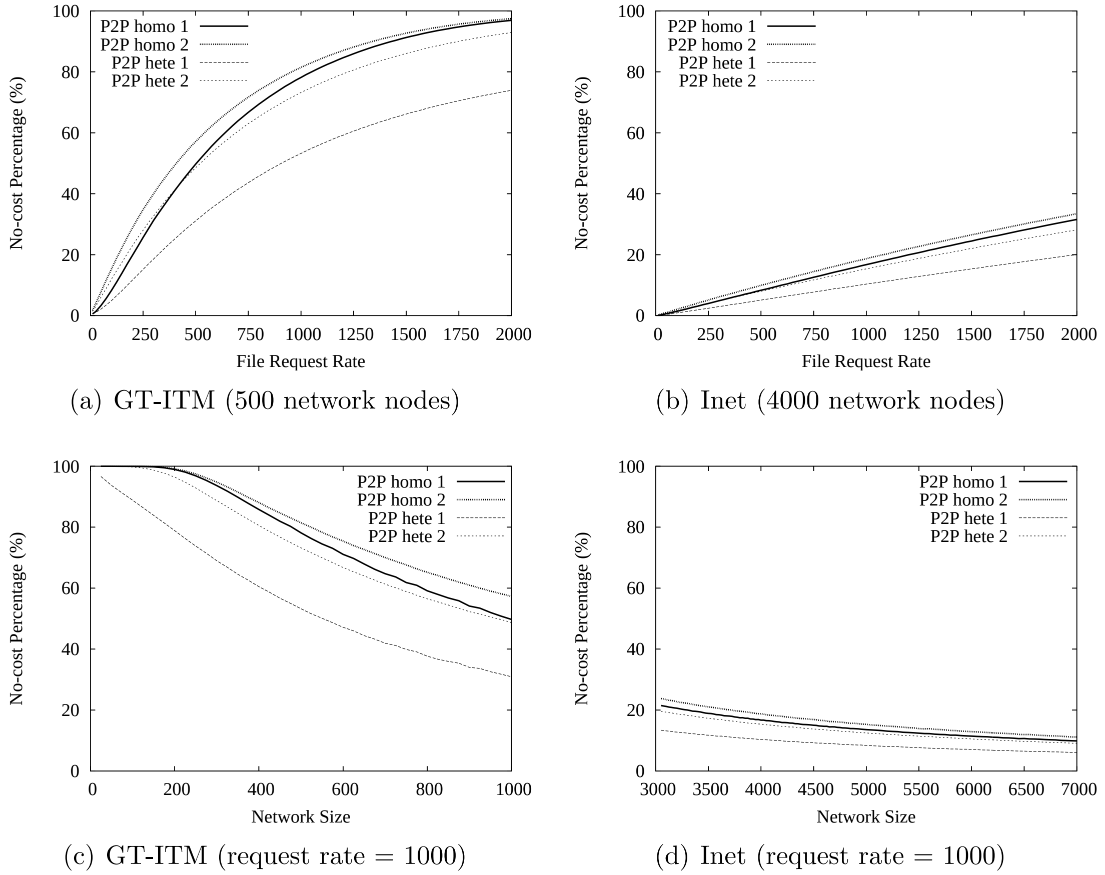


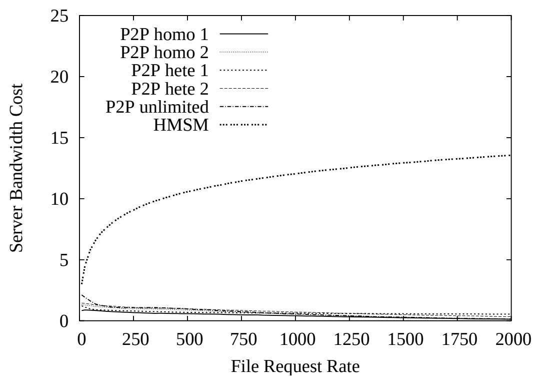
Figure 4.26: No-cost Percentage for Protocol with Limited Outbound Bandwidth

- When the heterogeneous setting is used, in spite of the performance variance between GT-ITM and Inet, the new protocol consumes more network bandwidth than that with the homogeneous setting. Compared with the new protocol in the homogeneous setting and the conventional HMSM, which are two extremes in terms of the workload balance, the new protocol with the heterogeneous setting achieves workload balance somewhere in-between and performs more like a multi-server version of the conventional HMSM, in which the majority of streams are delivered by a small number of peers who are assigned relatively high outbound bandwidth. Thus, in this case, there is a higher possibility for a peer, to listen to a stream delivered from another peer located at a different node, which causes lower no-cost percentage as shown in Figure 4.26, compared with the homogeneous setting. Figure 4.26 also shows that, as the limited capacity of heterogeneous outbound bandwidth increases from 1 to 2, more peers listen to no-cost streams, resulting in a visible reduction on the network bandwidth cost as observed from Figure 4.25.

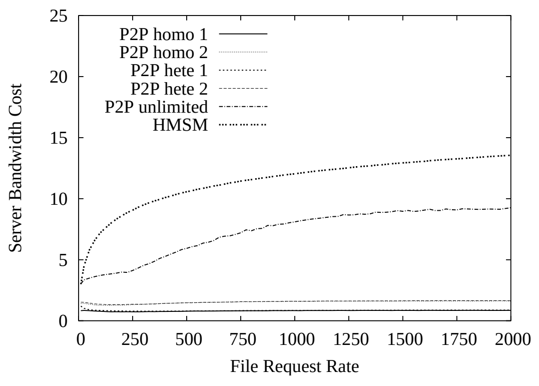
- Although the new protocol with the heterogeneous outbound bandwidth, which is more realistic, consumes more network bandwidth than it is with unlimited or homogeneous limited outbound bandwidth, results show, with a reasonable limited capacity of outbound bandwidth, it still performs better than the conventional HMSM, which is more meaningful to this performance study.

Figure 4.27 provides the server bandwidth cost for the new protocol with limited outbound bandwidth, in terms of the effects of file request rate and network size. Results show that the new protocol with limited outbound bandwidth, homogeneously or heterogeneously, achieves a significant reduction on the server bandwidth cost, compared with the conventional HMSM. Also, since the outbound bandwidth limitation implicitly leads to a better workload balance, the new protocol with the limitation is more steadily efficient in terms of the server bandwidth cost than it is with unlimited outbound bandwidth.

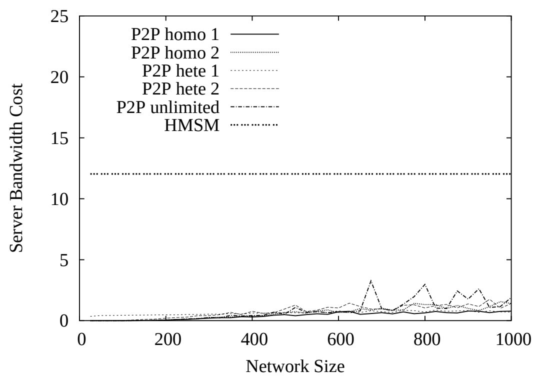
The effect of limited outbound bandwidth capacity is also studied in this section. Figure 4.28 provides the sample results of the network and server bandwidth costs for both homogeneous or heterogeneous settings. Two observations are presented as



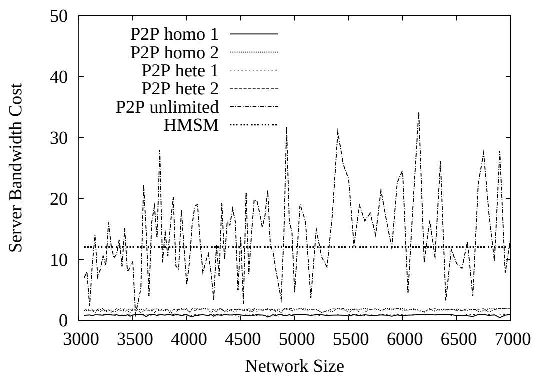
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)



(d) Inet (request rate = 1000)

Figure 4.27: Impact of Limited Outbound Bandwidth on Server Bandwidth Cost

follows:

- For the new protocol with homogeneous outbound bandwidth, the performance of the network/server bandwidth cost is constant in the range of outbound bandwidth capacities measured in simulation (i.e., 1 to 10), which means outbound bandwidth of 1 is sufficient to achieve good performance.

- For the new protocol with the heterogeneous setting in which the measured outbound bandwidth capacity could be less than 1, the network bandwidth cost shows a sharp decrease followed by a slow decrease, and finally stabilizes at the same value as that of the homogeneous setting; the server bandwidth cost reaches its critical point at around 0.4, which implies the outbound bandwidth average of 0.4 is sufficient for serving all peer requests and for the whole

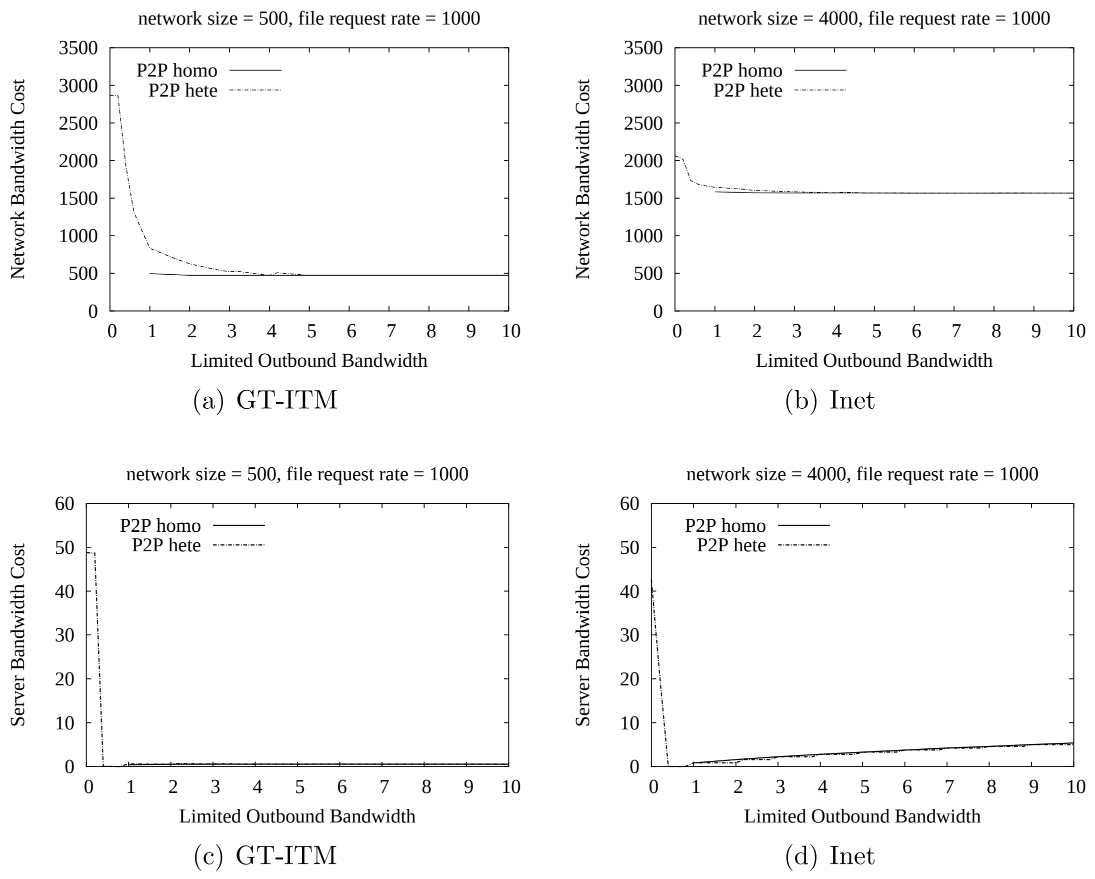


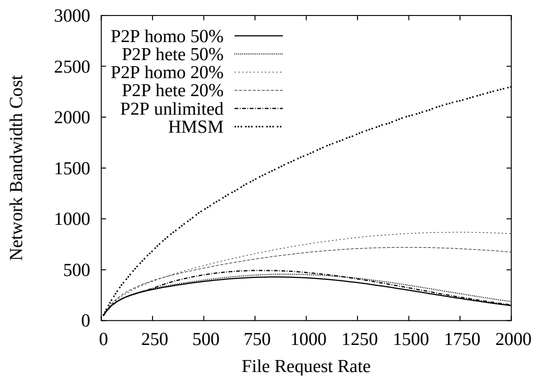
Figure 4.28: Effect of Limited Outbound Bandwidth Capacity on Network/Server Bandwidth Cost

system to be effective. Overall, the outbound bandwidth from 1 to 3 is an acceptable range, considering the trade-off between high outbound bandwidth requirement and low network/server bandwidth cost.

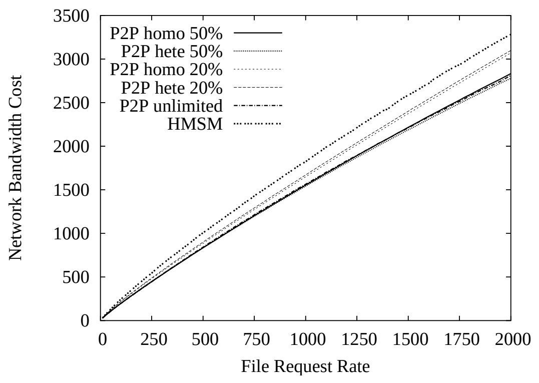
4.3.5 Impact of Limited Peer Buffer Space

In peer-to-peer stream merging, each peer needs local storage to buffer data even after its playback, which is not only for the attempted stream mergers but also for serving other peers. The results presented so far are derived from simulations that assume unlimited peer buffer space that enables a peer to deliver streams to any late-coming peers as long as it stays in the system, which is unlikely in practice. When the buffer space is more restrictive in the system, some peers may not be able to serve streams since the requested data is not available in the buffer due to a buffer overflow. In this section, the impact of such limitation on the performance reductions is studied.

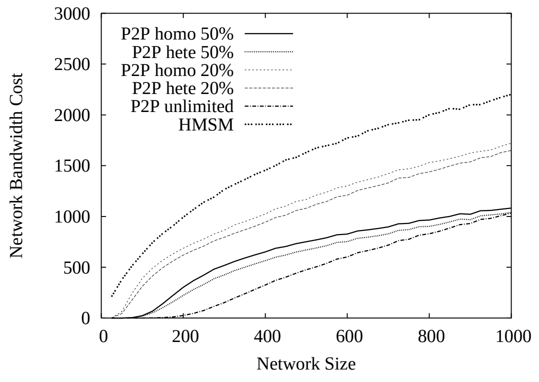
Similar to the study on the impact of limited peer outbound bandwidth, the impact of limited peer buffer space is studied using both the homogeneous and heterogeneous settings in simulations. With the homogeneous setting, all peers are assumed to have the same buffer capacity; and with the heterogeneous setting, the buffer capacities of all participating peers in the system are determined using the same method for the allocation of heterogeneous outbound bandwidth described in Section 4.3.4, i.e., the buffer capacity of a new peer is dynamically determined by comparing the averaged buffer capacity of all peers currently active in the system to the expected peer buffer capacity.



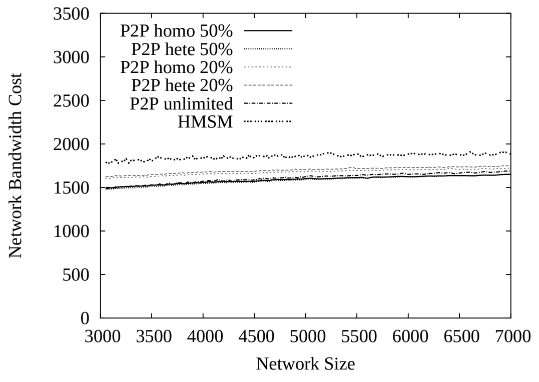
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)



(d) Inet (request rate = 1000)

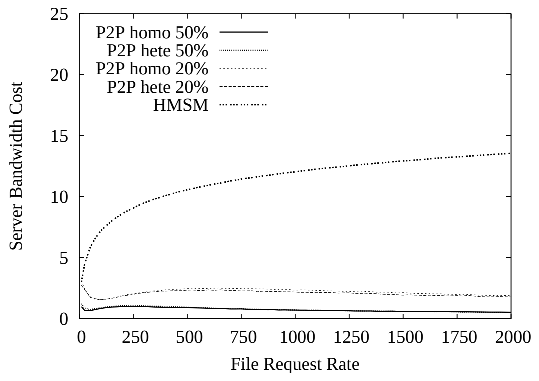
Figure 4.29: Impact of Limited Buffer Space on Network Bandwidth Cost

The default policies (i.e., the *location-first* policy and the extended stream merging policy) is used in the simulations that were conducted for the study on the impact of limited peer buffer space. Figure 4.29 provides the network bandwidth costs for the new protocol with limited and unlimited peer local storage and for the conventional HMSM as well, and illustrates how much reduction is induced by the limitation. In the figure, the results of the new protocol with homogeneous and heterogeneous buffer space are respectively labelled by “P2P homo” and “P2P hete” followed by a number that represents the corresponding averaged peer buffer capacity expressed in percentage of the full-file length. The lines labelled by “P2P unlimited” and “HMSM” represent results for the new protocol with unlimited buffer space and the conventional HMSM, respectively. Some observations obtained from the figure are described as follows:

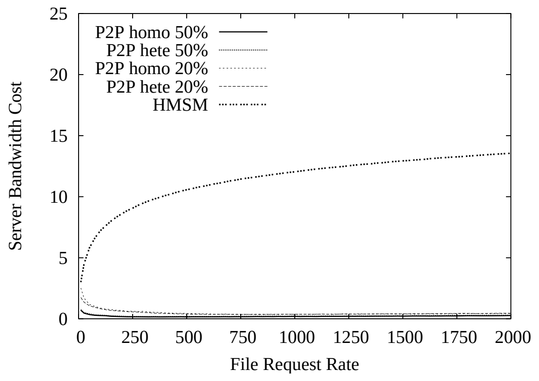
- Unlike the new protocol with the limited outbound bandwidth that presents distinct performance between the homogeneous and heterogeneous settings, the variance between the performance of homogeneous and heterogeneous peer buffer space is minor. As shown in the figure, when the buffer size is either 20% or 50% of the full-file length, the network bandwidth costs in both settings are close in both GT-ITM and Inet.

- No matter which of the two settings is used, the buffer size is the dominant factor that affects the protocol performance. Results show, in either case, the protocol consumes more network bandwidth when the buffer size is 20% of the file length than 50%. This is reasonable since, with less buffer space, more peers are not able to store data that may be useful later so that more new peers have to request data from the server or small number of peers who still have the requested data stored in their buffer, thus causing more network bandwidth consumed in delivery.

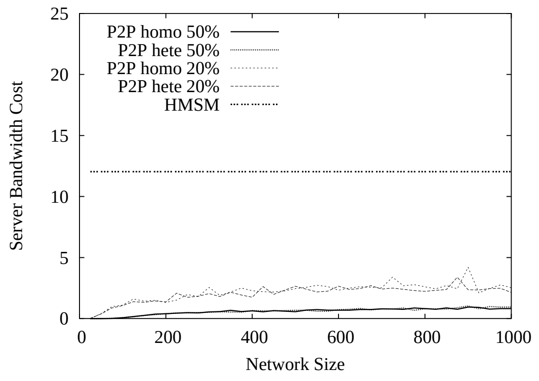
Figure 4.30 presents the comparison on the server bandwidth cost. As observed from the two sample results for buffer sizes set to 20% and 50% of the file length, in terms of the server bandwidth cost, the new protocol performs very similarly in both



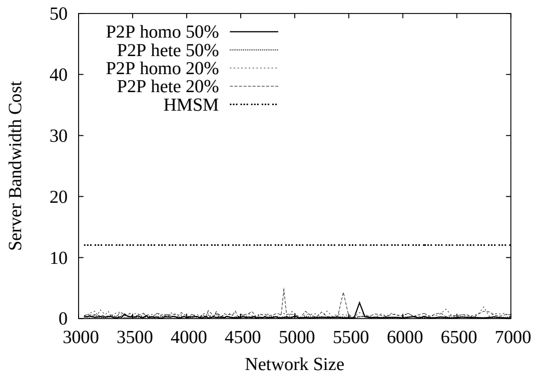
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)

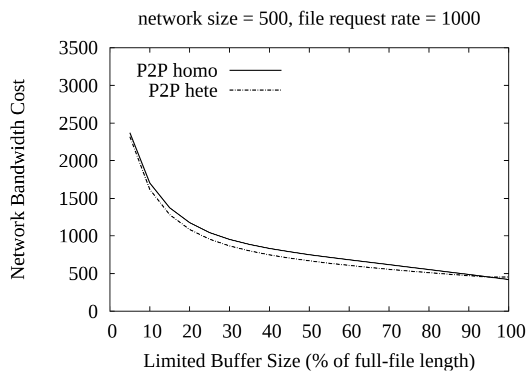


(d) Inet (request rate = 1000)

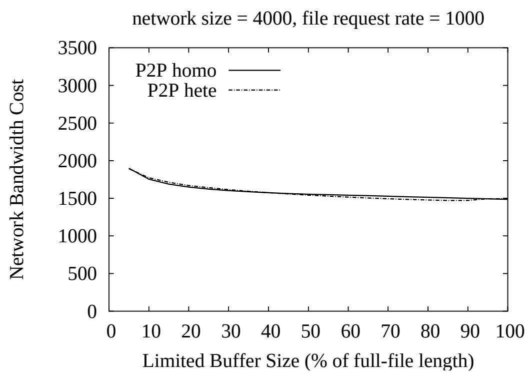
Figure 4.30: Impact of Limited Buffer Space on Server Bandwidth Cost

homogeneous and heterogeneous settings, and is more efficient than the conventional HMSM. Also it is expected that more server bandwidth will be consumed when peers have less buffer space, since the server alway stores the full-length file and is more possibly requested for data when fewer peers have the requested data stored in their buffers. It shows that the results conform to this expectation more clearly in the GT-ITM topology.

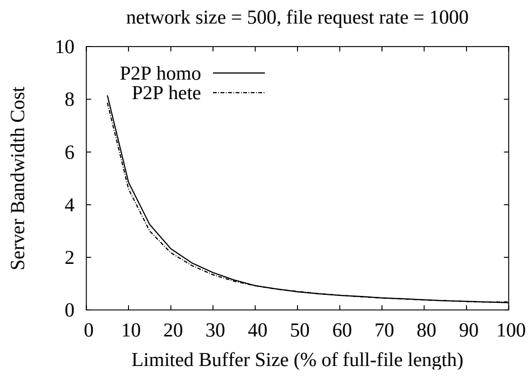
The scale of performance on the effect of buffer size is also studied. Figure 4.31 shows the network and server bandwidth costs in both the homogeneous and heterogeneous settings for buffer sizes in range of 5% to 100% of the full-file length, with the fixed file request rate and network size. Two observations are described as follows:



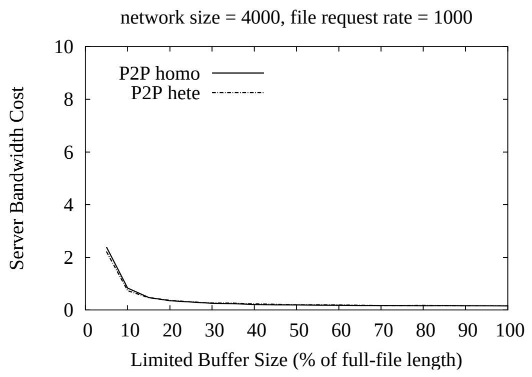
(a) GT-ITM



(b) Inet



(c) GT-ITM



(d) Inet

Figure 4.31: Effect of Limited Buffer Capacity on Network/Server Bandwidth Cost

- It shows again that the performance is nearly identical for the homogeneous and heterogeneous peer buffer space. As observed in the study on the impact of limited outbound bandwidth (Section 4.3.4), in which the protocol with the homogeneous setting performs better than that with the heterogeneous setting, a better-balanced workload helps to reduce the total network bandwidth consumed in media delivery. However, in the case of limited storage space, the workload balance is mostly affected by the buffer size regardless in which way, homogeneously or heterogeneously, the buffer capacity is allocated in the system.

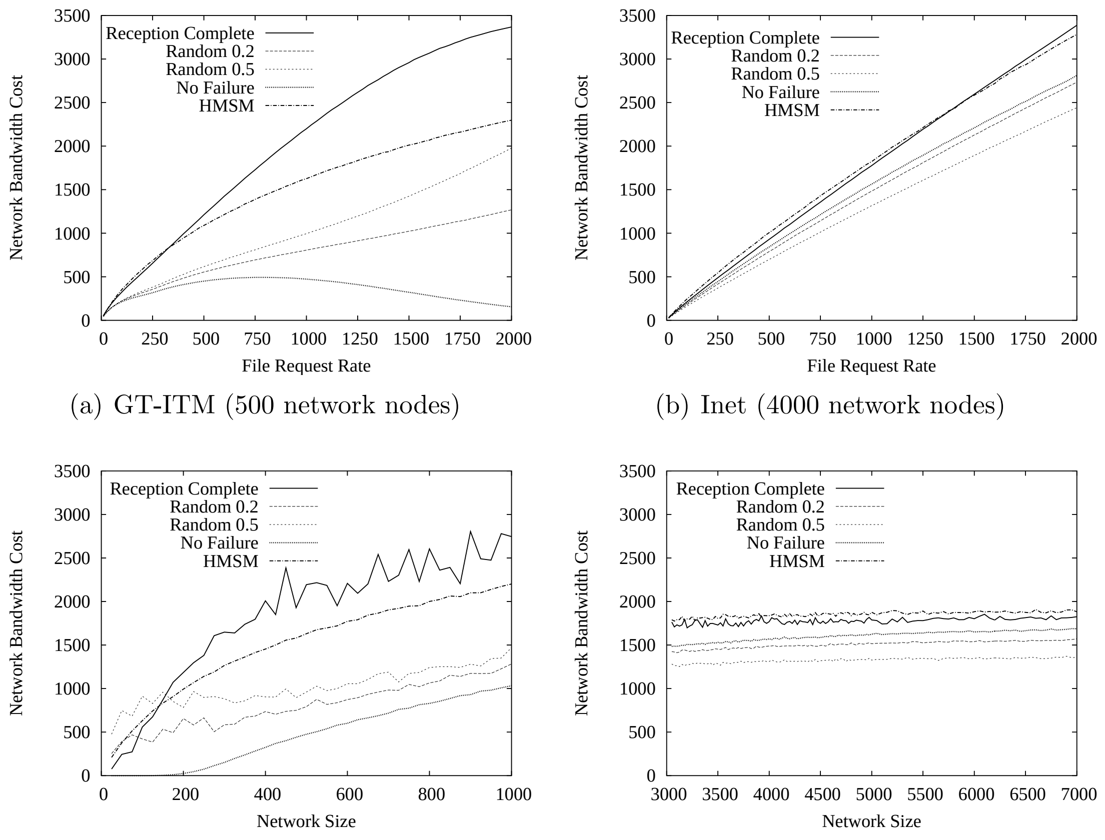
- Also observe that both the network and server bandwidth costs are extremely high when the buffer size is set to 5% of the full-file length. As the buffer size increases, both costs start with a sharp decrease and then slowly decline after the buffer size reaches approximately 10% or 20% of the file length (different in GT-ITM and Inet). Overall, the capability of storing 20% of the media file for each peer is sufficient for the new protocol to achieve performance competitive to that in the ideal case.

4.3.6 Stream Failure Recovery

Some important peer characteristics such as the short lifetime and the insufficient capability to be a dedicated server may easily cause a stream failure during a long streaming session. Section 3.5 introduces a recovery scheme that is designed to be receiver-driven and aims at recovering any stream failure caused by graceful and unexpected peer departures. This section evaluates the performance of this stream recovery scheme in terms of when the departure occurs. Specifically, two cases are studied: peer departure when receptions complete and peer departure at random.

Previous sections assume no stream failure in media delivery. In particular, to avoid stream failures caused by the peer departures when receptions complete, peers are assumed to stay in the system as long as they are still active in their uploading sessions even after their own receptions complete. However, this assumption is not valid in the simulation conducted for the first case of peer departure. Instead, each peer is assumed to leave the system as soon as its receptions complete. Note that, this is the extreme case of peer departure, i.e., all peers are assumed to leave, resulting in failures of all existing streams.

In the simulation conducted for the case of peer departure at random, the departing peers are chosen in random. Specifically, the failure events are designed to be evenly distributed over the simulation time. The time interval between two successive failure events depends on the failure rate which is defined as the number of peer departures divided by the number of active peers in the system. Since the number of active peers changes over time, the interval is dynamically updated whenever a new peer joins or an existing peer leaves gracefully or unexpectedly.



(a) GT-ITM (500 network nodes) (b) Inet (4000 network nodes)

(c) GT-ITM (request rate = 1000) (d) Inet (request rate = 1000)

Figure 4.32: Network Bandwidth Cost for Protocol with Stream Failure Recovery

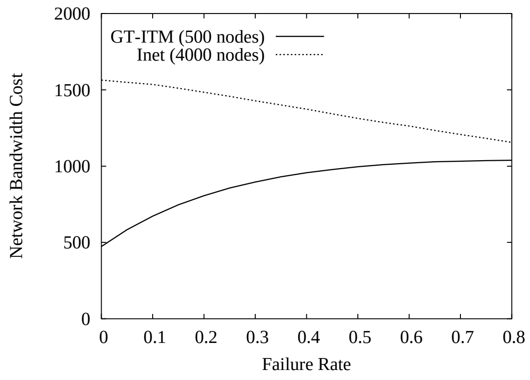
Figure 4.32 shows the network bandwidth costs for the new protocol in two cases of peer departures which are compared with the new protocol without failure and the conventional HMSM. In the figure, the results of protocol with two types of peer departures and without failure are respectively labelled by "Reception Complete", "Random", and "No Failure". The number following "Random" represents the failure rate. Some observations are discussed as follows:

- With the assumption of peer departure when receptions complete, the new protocol requires much more network bandwidth than that in the case of no stream failure, and it performs even worse than the conventional HMSM in GT-ITM. As shown in previous sections, the existence of "no-cost" streams allows the new protocol to be efficient in terms of the network bandwidth cost.

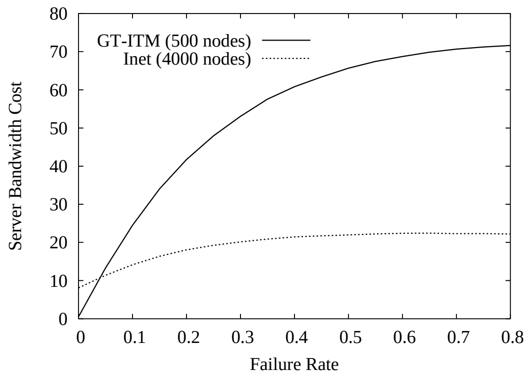
However, with the presence of peer departure when receptions complete, those "no-cost" streams, sooner or later, will fail and be replaced by recovery streams delivered by the server, which will certainly increase the network bandwidth cost.

- In the case of random departures, for the two sample failure rates 0.2 and 0.5, the new protocol consumes less network bandwidth than the conventional HMSM. However, two variances are present between GT-ITM and Inet: 1) compared with the protocol with no stream failure, the new protocol consumes more network bandwidth in GT-ITM, but less in Inet; 2) a higher failure rate causes more network bandwidth consumed in GT-ITM, but on the contrary in Inet. Note that, for any random departure, there are two possible effects on the total network bandwidth cost. If the departing peer is delivering one or more streams, the network bandwidth cost would possibly increase since more network bandwidth is required by the recovery stream; otherwise, the total network bandwidth cost would decrease since the data reception of the departing peer stops. Owing to the structure difference between GT-ITM and Inet, the dominance of which effect on the network bandwidth cost results in the above performance variances. The second performance variance is more clearly illustrated in Figure 4.33(a) which provides the network bandwidth costs assessed as function of failure rate in GT-ITM and Inet.

The server bandwidth cost is an important metric for evaluating the new protocol when stream failure is assumed. Figure 4.34 presents the server bandwidth costs assessed as functions of file request rate and network size, and Figure 4.33(b) shows the results assessed as function of failure rate for the case of random departure. Results show, the server bandwidth cost is extremely high in the case of peer departure when receptions complete, especially at high file request rate. This is not surprising since the case studied is the extreme case of peer departure, in which hundreds or even thousands of active streams will fail, requiring the server to initiate and deliver recovery streams. For the case of random departure, although the new protocol

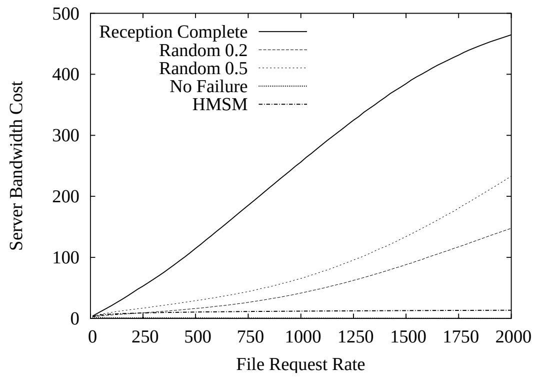


(a) Network Bandwidth Cost

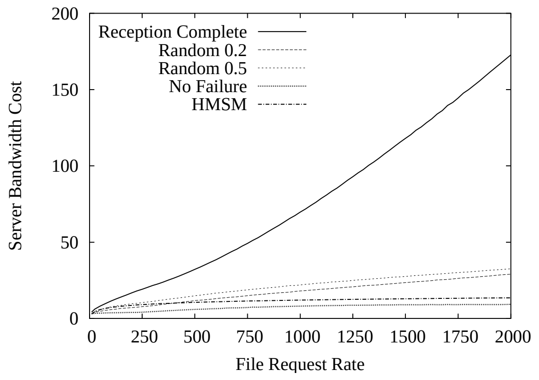


(b) Server Bandwidth Cost

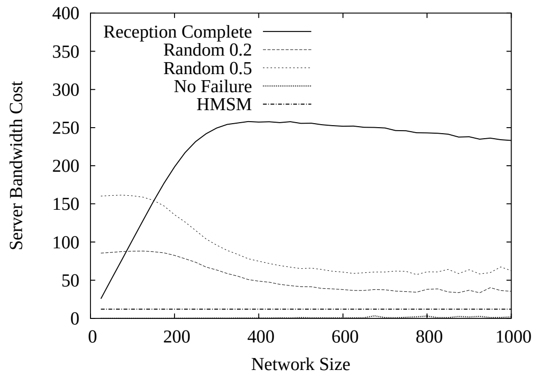
Figure 4.33: Effect of Peer Failure Rate on Network/Server Bandwidth Cost for the Case of Random Departure (file request rate = 1000)



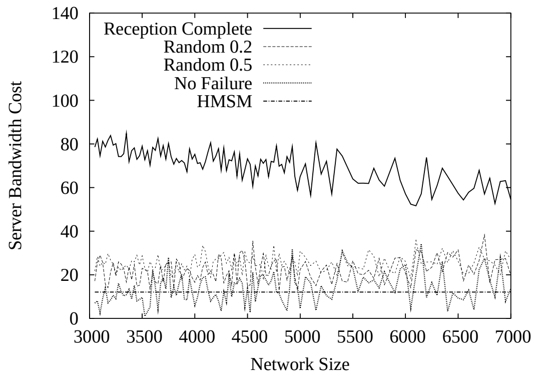
(a) GT-ITM (500 network nodes)



(b) Inet (4000 network nodes)



(c) GT-ITM (request rate = 1000)



(d) Inet (request rate = 1000)

Figure 4.34: Server Bandwidth Cost for Protocol with Stream Failure Recovery

performs better than that in the case of peer departure when reception complete, it still consumes more server bandwidth than the new protocol with no stream failure and the conventional HMSM, especially in GT-ITM.

Overall, although having the server handle stream recovery is the simplest and most robust option, the new protocol yields unsatisfactory performance and can not cope with a high rate of peer departures due to server overloading. For the protocol to be effective in real system, further changes could be considered to reduce the server bandwidth cost and hopefully the network bandwidth cost as well. The following briefly discusses two possible options that could be evaluated in the future work:

- One possible change is regarding the selection of the recovery stream initiators. Instead of the server only, the initiators could be peers which are selected based on the *location-first* or *location-bandwidth ratio* policies so that the server workload can be greatly reduced. Since this change requires additional time for the initiator selection, alternative methods can be applied to reduce the recovery time, e.g., adding a small start-up delay before the playback or increasing the stream delivery bit rate.

- Another possible change is to restrict the selection of stream initiators. In previous simulations that assume no peer departure, a peer could be chosen to deliver streams regardless of its own receptions. Considering that the recovery workload can be reduced if the remaining duration of a failed stream is short, a threshold on data reception can be used to restrict the initiator selection, i.e., only peers whose data receptions are less than the threshold are qualified to be candidates of a stream initiator.

4.3.7 Summary

Previous sections describe a number of simulations that have been done in the performance study. Based on the obtained results, a brief summary is presented in this section. Basically, the performance of the peer-to-peer stream merging protocol is evaluated under two different assumptions: unlimited and limited peer resource capabilities. The remainder of this section separately summarizes the simulations under these two categories.

Unlimited Peer Resource Capability

With the assumption of unlimited peer outbound bandwidth and unlimited buffer space, four sets of simulations are performed as summarized as follows:

- The new protocol with the default policy and parameter settings is compared with the conventional HMSM protocol. Although there is performance variance between GT-ITM and Inet, overall the new protocol yields better performance than the conventional HMSM, in terms of both the network and server bandwidth costs.
- The performance of the three peer selection policies is compared. Among the three policies, the *location-bandwidth ratio* policy yields the best performance. It achieves almost the same efficiency on the network bandwidth cost as the *location-first* policy, and meanwhile, the overall system is well balanced and at most two streams are delivered by a peer or the server. The *location-first* policy is efficient on the network bandwidth cost but its performance on the server bandwidth cost is not stable. For the *load-first* policy, it is the best option only when the system workload balance is concerned.
- The extended stream merging policy is compared with the conventional early merging policy when both are applied in the new protocol and combined with the *location-first* policy. In terms of the network bandwidth cost, the new protocol with the early merging policy is less efficient than that with the extended stream merging policy, and performs very closely to the conventional HMSM. This implies that, besides the peer selection policy, the stream merging policy is also a key factor that affects the network bandwidth cost.
- The proposed stream recovery scheme is evaluated in two cases of peer departures. Owing to the server overloading problem, the performance is unsatis-

factory. For the recovery scheme to be effective in large-scale systems with a high early peer departure rate, modifications need to be made in the future research.

Limited Peer Resource Capability

Two sets of simulations are performed to study the impacts of limited peer outbound bandwidth and limited peer buffer space, respectively. The following summarizes these simulations:

- The first set of simulation evaluates the impacts of both the homogeneous and heterogeneous limited peer outbound bandwidths on the performance reductions. Results show, if the same capacity is used, the protocol with the homogeneous setting generally consumes less network bandwidth than that with the heterogeneous setting, which implies that better workload balance helps to reduce the network bandwidth cost. When the system is more realistically set with the heterogeneous outbound bandwidth, the outbound bandwidth average from 1 to 3 is an acceptable range for the protocol to achieve reasonably good performance, considering the trade-off between high outbound bandwidth requirement and low network/server bandwidth cost.

- The second set of simulation evaluates the impact of limited peer buffer space, also with both the homogeneous and heterogeneous settings. Results show, the performance is mostly affected by the amount of buffer capacity rather than the type of buffer space setting. Overall, for both settings, the buffer capacity of 10% or 20% of the file length (different in GT-ITM and Inet) is sufficient for the new protocol to achieve relatively good performance.

Chapter 5

Conclusions

To achieve a more scalable VOD system design that enables efficient concurrent service to large numbers of wide-area clients, this thesis develops a streaming protocol that adapts the previously proposed HMSM protocol to use a peer-to-peer delivery approach. To conclude the work that has been done in this thesis including the protocol design and the simulation-based performance study, this chapter provides a brief summary of the thesis, states the contributions, and presents some possible future research directions.

5.1 Thesis Summary

The first two chapters of the thesis provide a brief introduction and a literature review of VOD streaming. Various previously proposed techniques for scalable VOD streaming are reviewed, and the corresponding constraints and limitations of each technique are summarized.

Chapter 3 develops a new VOD streaming protocol that adapts a previously proposed stream merging protocol (i.e., HMSM) for use in a peer-to-peer system. The overview of the new protocol is first provided including the system architecture, the stream merging structure, and some important peer and server characteristics. The two key components of the new protocol, the selection policy for the primary stream initiator and the stream merging policy, are then discussed in detail. Three policies are proposed for selection of the primary stream initiator that differ according to how they take into account peer location and workload, and a new stream merging policy is proposed by adapting the conventional early merging policies to the peer-to-peer context. Lastly, approaches for dealing with peer storage limitations and the stream disruptions caused by peer departures are discussed.

Chapter 4 presents the performance evaluation of the new protocol. Based on a number of assumptions and the simulation methodology that are described at the beginning, six sets of simulations are conducted. First, the performance of the new protocol with the default policy and parameter settings is compared with that of the conventional HMSM. The performance of the three peer selection policies is then studied and compared. The performance of the extended stream merging policy is also compared with that of the conventional early merging policy. Furthermore, the impacts of limited peer outbound bandwidth and limited peer buffer space on the overall performance are studied. Finally, a performance study of the proposed stream recovery scheme is presented.

5.2 Contributions

This work aims at improving the scalability of a VOD streaming system. In summary, the main contributions of the thesis are as follows:

- The key contribution of this thesis is the development and evaluation of a scalable VOD streaming protocol that enriches the stream merging technique by adapting the conventional HMSM protocol to the peer-to-peer context. It improves the efficiency of previously proposed stream merging protocols in terms of both the server and network bandwidth requirements.

- Three peer selection policies are developed and evaluated. The *location-bandwidth ratio* policy is efficient in terms of the network and server bandwidth costs, and meanwhile, achieves the system workload balance as well.

- A new stream merging policy is developed that adapts the conventional early merging policies to provide efficient stream merging in the peer-to-peer context.

- The impacts on performance of limited peer outbound bandwidth and limited client buffer space are studied, both for homogeneous and heterogeneous peer scenarios. The new protocol with limited peer outbound bandwidth is more efficient when the workload is balanced, and the buffer size of about 10% or 20% of the file length (different in GT-ITM and Inet) is sufficient for the new protocol with limited peer buffer space to achieve relatively good performance.
- A scheme of stream failure recovery is developed to handle stream failures caused by any type of peer departure.

5.3 Future Work

This thesis studies some important issues related to scalable VOD streaming. This section discusses some possible directions that can be investigated following this work in future research:

- Modifying the stream recovery scheme so that it will be effective and efficient in large-scale systems with a substantial early peer departure rate. Results show that the stream recovery scheme proposed in this work can not effectively cope with a high rate of early peer departures due to server overloading. As mentioned in Section 4.3.6, several modifications are possible to improve the efficiency in terms of the server bandwidth cost and likely the network bandwidth cost.
- Improving the design of the peer-to-peer stream merging protocol and implementing a prototype to support experiments in large scale real systems. In this thesis, some details of the protocol design are simplified since the performance evaluation of this work is simulation-based. One example is the method for measuring network distance which needs to be determined for real-system

experiments. Another example concerns the network bottleneck between two nodes which in the simulation experiments is simply assumed to be at an end point rather than at some intermediate point on the communication path. To more precisely measure the bottleneck and the related end-to-end available bandwidth between two peers, a more specific method needs to be included in the design and implemented in the prototype. Moreover, another possible future work direction is the implementation of suitable application-layer multicast support for the protocol prototype, as motivated by the lack of reliable wide-area IP multicast service in the current Internet.

- If a prototype is implemented in future work, experiments could be conducted in LAN environments as well as over the Internet. This thesis uses a simple simulator that is based on a number of assumptions. For experiments in a real system, many other issues are of concern, for instance, the impact of the delivery latency on the media playback, the control overhead of the protocol, and the real-system evaluation of the stream recovery scheme. For such issues, additional performance metrics can be used besides the three bandwidth costs. For example, the time required for recovering a failed stream can be another metric for the evaluation of the stream recovery scheme.

- Providing support for interactive functions such as pause, rewind, and fast-forward, which is essential for the peer-to-peer stream merging protocol to be used in VOD streaming applications. Since peers generally have storage limitations in real systems, it is not trivial to design these interactive functions. New protocol techniques may be required when newly requested data is not available in the buffer, in which case additional overhead is probably exposed.

References

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems. In *Proc. IEEE ICMCS*, pages 118–126, Hiroshima, Japan, June 1996.

[2] J. M. Almeida, D. L. Eager, M. C. Ferris, and M. K. Vernon. Provisioning Content Distribution Networks for Streaming Media. In *Proc. IEEE INFOCOM 2002*, volume 3, pages 1746–1755, New York, NY, June 2002.

[3] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. Wright. Minimizing Delivery Cost in Scalable Streaming Content Distribution Systems. *IEEE Trans. on Multimedia*, 6(2):356–365, Apr. 2004. Special Issue on Streaming Media.

[4] K. C. Almeroth. The Evolution of Multicast: From the MBone to Inter-Domain Multicast to Internet2 Deployment. *IEEE Network Special Issue on Multicasting*, 14(1):10–20, Jan./Feb. 2000.

[5] S. Annapureddy, C. Gkantsidis, and P. Rodriguez. Providing Video-on-Demand using Peer-to-Peer Networks. In *Proc. Internet Protocol TeleVision (IPTV) workshop, WWW'06*, Edinburgh, Scotland, United Kingdom, May 2006.

[6] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. ACM Sigcomm 2002*, pages 205–217, Pittsburgh, Pennsylvania, Aug. 2002.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Content Distribution in a Cooperative Environment. In *Proc. IPTPS'03*, volume 2735, pages 292–303, Berkeley, CA, USA, Feb. 2003.

[8] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communication (JSAC)*, 20(8):1489–1499, Oct. 2002.

[9] Y. Chawathe, S. McCanne, and E. Brewer. An Architecture for Internet Content Distribution as an Infrastructure Service. Unpublished, 2000.

[10] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A Case for End System Multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471, Oct. 2002.

[11] Y. Cui and K. Nahrstedt. Layered Peer-to-Peer Streaming. In *Proc. International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '03)*, pages 162–171, Monterey, CA, June 2003.

[12] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proc. ACM SIGCOMM'04*, pages 15–26, Portland, OR, Aug. 2004.

[13] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The PIM Architecture for Wide Area Multicasting. *IEEE/ACM Transactions on Networking*, 4(2):153–162, Apr. 1996.

[14] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming Live Media over Peers. Technical Report 2002-21, Stanford University, Mar. 2002.

[15] T. T. Do, K. A. Hua, and M. A. Tantaoui. P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment. In *Proc. IEEE International Conference on Communications (ICC'04)*, volume 3, pages 1467–1472, Paris, June 2004.

[16] D. L. Eager, M. K. Vernon, and J. Zahorjan. Optimal and Efficient Merging Schedules for Video-on-Demand Servers. In *Proc. 7th ACM International Multimedia Conference (ACM MULTIMEDIA'99)*, pages 199–202, Orlando, FL, Oct. 1999.

[17] D. L. Eager, M. K. Vernon, and J. Zahorjan. Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand. In *Proc. SPIE Multimedia Computing and Networking (MMCN'00)*, pages 206–215, San Jose, CA, Jan. 2000.

[18] D. L. Eager, M. K. Vernon, and J. Zahorjan. Minimizing Bandwidth Requirements for On-Demand Data Delivery. *IEEE Transactions On Knowledge and Data Engineering*, 13(5):742–757, Sep./Oct. 2001.

[19] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y.Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Trans. on Networking*, 9(5):525–540, Oct. 2001.

[20] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proc. ACM SIGCOMM IMW 2002*, pages 5–18, Marseilles, France, Nov. 2002.

[21] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2Cast: Peer-to-Peer Patching Scheme for VoD Service. In *Proc. the 12th World Wide Web Conference (WWW-03)*, pages 301–309, Budapest, Hungary, May 2003.

[22] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. PROMISE: Peer-to-Peer Media Streaming Using CollectCast. In *Proc. ACM Multimedia '03*, pages 45–54, Berkeley, CA, Nov. 2003.

[23] A. Hu. Video-on-Demand Broadcasting Protocols: A Comprehensive Study. In *Proc. IEEE Infocom*, volume 1, pages 508–517, Anchorage, AK, Apr. 2001.

[24] N. Hu and P. Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE JSAC*, 21(6):879–894, Aug. 2003.

[25] K. A. Hua, Y. Cai, and S. Sheu. Patching: A Multicast Technique for True Video-on-Demand Services. In *Proc. ACM Multimedia'98*, pages 191–200, Bristol, U.K., Sept. 1998.

[26] K. A. Hua and S. Sheu. Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems. In *Proc. ACM SIGCOMM 97*, pages 89–100, Cannes, France, Sept. 1997.

[27] M. Jain and C. Dovrolis. End-to-End Available Bandwidth: Measurment Methodology, Dynamics and Relation with TCP Throughput. In *Proc. ACM SIGCOMM'02*, pages 295–308, Pittsburgh, PA, USA, Aug. 2002.

[28] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proc. 4th Symp. Operating System Design and Implementation (OSDI 2000)*, pages 197–212, San Diego, California, Oct. 2000.

[29] S. Jin and A. Bestavros. Cache-and-Relay Streaming Media Delivery for Asynchronous Clients. In *Proc. NGC 2002*, Boston, Massachusetts, USA, Oct. 2002.

[30] L. Juhn and L. Tseng. Harmonic Broadcasting for Video-on-Demand Service. *IEEE Transactions on Broadcasting*, 43(3):268–271, Sept. 1997.

[31] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin, and M. Handley. The MASC/BGMP Architecture for Inter-Domain Multicast Routing. In *Proc. ACM SIGCOMM'98*, pages 93–104, Vancouver, BC, Canada, Aug. 1998.

[32] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Addison-Wesley, 2nd edition, 2002.

[33] K. Lakshminarayanan, V. Padmanabhan, and J. Padhye. Bandwidth Estimation in Broadband Access Networks. In *Proc. ACM IMC'04*, pages 314–321, Taormina, Italy, Oct. 2004.

[34] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-Driven Layered Multicast. In *Proc. SIGCOMM'96*, pages 117–130, Stanford, CA, Aug. 1996.

[35] T. S. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proc. IEEE INFOCOM 2002*, volume 1, pages 170–179, NewYork, NY, June 2002.

[36] J.-F. Paris, S. W. Carter, and D. D. E. Long. A Low Bandwidth Broadcasting Protocol for Video on Demand. In *Proc. IEEE Int'l Conference on Computer Communications and Networks*, pages 690–697, Lafayette, Lousiana, Oct. 1998.

[37] J.-F. Paris, S.W. Carter, and D. D. E. Long. A Hybrid Broadcasting Protocol for Video on Demand. In *Proc. 1999 Multimedia Computing and Networking Conference (MMCN'99)*, pages 317–326, San Jose, CA, Jan. 1999.

[38] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, pages 329–350, Heidelberg, Germany, Nov. 2001.

[39] L. H. Sahasrabuddhe and B. Mukherjee. Multicast Routing Algorithms and Protocols: A Tutorial. *IEEE Network*, 14(1):90–102, Jan./Feb. 2000.

[40] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.

[41] J. Strauss, D. Katabi, and F. Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. In *Proc. ACM SIGCOMM Internet Measurement Conference '03*, pages 39–44, Miami, Florida, Oct. 2003.

[42] D. A. Tran, K. A. Hua, and T. T. Do. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In *Proc. IEEE INFOCOM '03*, volume 2, pages 1283–1292, San Francisco, CA, USA, Mar.-Apr. 2003.

[43] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-Like Congestion Control for Layered Multicast Data Transfer. In *Proc. IEEE INFOCOM'98*, volume 3, pages 996–1003, San Francisco, CA, USA, Mar. 1998.

[44] S. Viswanathan and T. Imielinski. Pyramid Broadcasting for Video on Demand Service. In *Proc. IEEE Multimedia Computing and Networking Conference*, pages 66–77, San Jose, CA, 1995.

[45] D. Waitzman, S. Deering, and C. Partridge. Distance Vector Multicast Routing Protocol (DVMRP). RFC 1075, Nov. 1988.

[46] B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. In *Proc. IEEE INFOCOM 2002*, volume 3, pages 1726–1735, New York, NY, June 2002.

[47] J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, Department of EECS, University of Michigan, 2002.

[48] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. IEEE INFOCOM '96*, volume 2, pages 594–602, San Francisco, CA, Apr. 1996.

[49] X. Zhang, J. Liu, B. Li, and T. P. Yum. CoolStreaming/DONet: a Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming. In *Proc. IEEE INFOCOM'05*, volume 3, pages 2102–2111, Miami, FL, Mar. 2005.

[50] Y. Zhao, D. L. Eager, and M. K. Vernon. Network Bandwidth Requirements for Scalable On-Demand Streaming. In *Proc. IEEE INFOCOM 2002*, volume 1, pages 1119–1128, New York, NY, June 2002.