國立交通大學

資訊科學與工程研究所

碩 士 論 文

一個在同儕式網狀串流系統中之

有效的來源選擇方法

An Efficient Source Peer Selection Approach in P2P Mesh-Based Streaming System

研 究 生：陳冠騰

指導教授：陳耀宗　教授

中 華 民 國　九 十 八　年 七 月

一個在同儕式網狀串流系統中之有效的來源選擇方法

An Efficient Source Peer Selection Approach in P2P Mesh-Based Streaming System

研 究 生：陳冠騰　　Student：Kuan-Teng Chen

指導教授：陳耀宗　　Advisor：Yaw-Chung Chen

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

Department of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

一個在同儕式網狀串流系統中之有效的來源選擇方法

學生：陳冠騰　　指導教授：陳耀宗 博士

國立交通大學資訊科學與工程研究所

摘要

目前大部分的網際網路服務皆採用傳統的用戶伺服器(client-server)模式，隨著網際網路的頻寬不斷擴展增加，需要較高網路頻寬的服務愈來愈被廣泛使用，例如，網路電話(Voice over Internet Protocol)、隨選視訊(Video on Demand)、網路電視(Internet Protocol Television)。然而，當使用者數目劇增時，用戶伺服器模式將遭遇到可容納度及高成本的問題。而在所有這些應用服務中，網路電視又是占用最多資源的。例如，頻寬、儲存空間，而且可能遭遇大量使用者同時使用所導致資源不足的問題。為了解決系統資源不足的問題，在近十年，發展出一種熱門的網路服務架構：同儕式網路(Peer-to-Peer Network)，簡稱 P2P 網路，越來越多的網路電視利用同儕式架構以提供服務。而因為 P2P 網路中，排程方法(Scheduling Method)影響到來源端到目的端所花的時間，每條連線的頻寬都不盡相同，如何有效的利用不同的頻寬，使得資料的下載時間可以縮短，讓使用者更快的取得所要的資料，以增進服務品質，是設計 P2P 排程方法的一大重點。

在本篇論文中，我們介紹一些現有的排程方法，並提出一個來源同儕(source peer)的選擇方法，利用一些探測封包，探測所有同伴(partner)的可用頻寬以及傳輸狀態，利用這些數值選擇出一個最佳的來源並從其取得資料，有效的利用狀態最佳的同儕，降低取得資料的時間。最後利用 OMNet++模擬程式來驗證所提出的方法。

An Efficient Source Selection Approach in P2P Mesh-Based Streaming System

Student：Kuan-Teng Chen Advisor：Dr. Yaw-Chung Chen

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

Many popular applications and services currently deployed on Internet are based on the traditional client-server model. As the internet steadily broadens the bandwidth scope, some application services such as VoIP (Voice over Internet Protocol), VoD (Video on Demand), IPTV (Internet Protocol Television) that required high bandwidth are deployed extensively. Although Client-Server model is intuitive and easy for implementation, however, the server will become a potential bottleneck when the number of users grows. It is always difficult for a server to provide the service to a large number of clients at the same time because of high cost and scalability problem. Among these services IPTV needs the largest amount of resources such as bandwidth and storage, and it may suffer shortage of resources when a certain number of users use the system at the same time. To accommodate system resource problem, a popular architecture - Peer-to-Peer (P2P) network for Internet service has been developed in recent decade, more and more IPTV services are deployed based on P2P architecture. In P2P network, data scheduling algorithm has a major effect on the data transfer time between the source and destination. Since every link has a different bandwidth, so how to efficiently use the different bandwidth to reduce the data downloading time so that users can get the data quickly becomes an important issue for designing the P2P scheduling algorithm.

In this thesis, we proposed an efficient source peer selection algorithm which uses

probing packets to detect both the available bandwidth and transmission status of all of its partners, and use these statistics to choose the current best source to retrieve data so that the download time can be minimized. Finally, OMNet++ simulation shows that our proposed approach can significantly reduce the downloading time with minimum overhead.

Acknowledgement

First of all, I would like to express my sincerity to my advisor Prof. Yaw-Chung Chen for leading me into research area and enlarge my sight. In addition, Prof. Yaw-Chung Chen also provides me many useful suggestions and advices on the writing of this thesis duration.

Besides, I appreciate plenty to the members of Multimedia Communication Laboratory, including Yuan, AGA, REX, estar, ydoon, bestkid, kummon98, Ying-Chieh, Shang-Shu Sharon, and all of my friends for their cheers and encouragement. And especially for Cheryl Lin, without her help, I can't finish my thesis more quickly.

Finally, I would like to express my appreciation to my parents, my family, without their support, I won't be with current status.

Contents

Table list

Figure list

Chapter 1 Introduction

Peer-to-Peer network has attracted tremendous attention from both academy and industry. Nodes communicate directly with each other for the sharing and exchanging of data as well as other resources. In this chapter, we give the overview of the P2P system and our motivation

1.1 Overview

Since the Internet was started early in 1970s, many services and applications have been deployed, these include WWW, E-mail, File Transfer Protocol (FTP) and BBS. These services are all based on Client-Server model, on which the server is placed somewhere on Internet and waiting for client's request. The server starts to provide its service to client upon receiving client's request.

Client-Server model is easy for implementation. However, when the system population grows up, the server always becomes the bottleneck because of the limitation in centralized resources. So far, it is still hard for a single server to provide service to a large number of clients at the same time.

In the past decade, the so-called Peer-to-Peer (P2P) network architecture emerged. In this architecture, every peer acts as both server and client. A peer provides and receives the resource from/to other peers. This is the main difference between P2P architecture and Client-Server architecture, there are many advantages on P2P architecture, and it can resolve the server bottleneck and efficiently make a good use for every link.

The early application on P2P architecture is file-sharing. Every peer can share its own file among peers. Applications of Peer-to-Peer have been the focus point in recent years with the widely use of Napster [1], which is a file sharing system for sharing music files. To improve the file sharing performance on P2P architecture, BitTorrent (BT) [2] was developed

in 2002 by Bram Cohen which is a new architecture based on P2P file-sharing. To share a file through BitTorrent, a torrent file is first created. The torrent file contains the information of a so-called tracker that coordinates the file distribution and the metadata of the shared content. The file content is divided into many fixed-size chunks so that a peer can download a file from different peers for different chunks simultaneously. This can improve the download rate much faster. The concept of "chunk" causes great effect not only on file-sharing but also on many applications.

In addition to file-sharing, there were many applications deployed based on P2P architecture, such as VoIP(Voice over Internet Protocol). Skype [3] is probably the most well known VoIP service. It runs under proprietary protocol and relies on the Skype P2P network for user directory and firewall/NAT(network address translator) traversal. As a result, Skype can bypass the costly infrastructure associated with the deployment of VoIP, and can easily scale to a very large size.

Another popular application is P2P streaming. More than dozen companies are actively working in this area. Some example companies are PPLive [4] , PPStream [5], etc. The main reason that the IPTV runs on P2P architecture is scalability and provisioning cost, it can utilize every peer's upload bandwidth to provide contents to other peers. Thus it can reduce the workload in centralized server.

The source of video content is always divided into chunks in P2P systems. Peers receive chunks from source and share it with other peers. To achieve efficient and robust P2P content delivery, we need to work on two primary functions: P2P overlay construction and P2P scheduling. There are two major P2P overlays: Tree overlay and Mesh overlay. Every peer is a node in the tree overlay and it received video stream from parent and delivered contents to its children. Peer receives and delivers chunks from/to uncertain peers in mesh overlay. The mesh overlay is more robust compared to the tree overlay and it can accommodate much more peers than tree overlay. However, its latency is much longer than that for tree overlay.

Although there are many P2P IPTV system developed in recent years, most of them are proprietary systems. They do not release their protocol and algorithm in their systems. Many researchers are still curious about the internal design of these systems. Works have been done by tracing their packets [6] in the system and we can have deep view into these systems.

1.2 Motivation and Purpose

The P2P scheduling concerns the method for delivering the data from the source to its destinations under a given overlay. Depending on whether the sender and/or the receiver take the initiative in requesting the blocks, three modes may be used in the P2P delivery. In the push mode, the sender takes the initiative. In the pull mode, the receiver takes the initiative. In hybrid mode, both of them may take the initiative. In either push or pull mode, the peer needs to decide upon a delivery strategy, in the other word, which block to push/pull, and from/to which peer that the block is pushed and pulled. We call this the block selection and the peer selection [7].

Nowadays, many researchers focus on block selection, and proposed many block selection schemes, such as rarest, random etc. However only few of them focus on peer selection, they just randomly choose the peer who has the block, and receive the block from it. Because every link has a different bandwidth, it may result a longer download time if we choose the worse one.

To efficiently utilize the bandwidth resource available in the P2P network, we propose a new approach to choose the best peer among the current partners. It can cope with any other block selection schemes and achieve much better stability. We focus on mesh-based system due to its good scalability and ease in implementation. Our goal is to shorten the download time of the chunks in P2P streaming system so that user can have a better quality-of-experience (QoE) for using streaming services.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows. We introduce P2P streaming systems in more detail and address some related works about our peer selection algorithm in Chapter 2. In Chapter 3, we discuss our proposed scheme in detail. We evaluate our scheme through OMNet++ simulation in Chapter 4. Finally, conclusions are stated in Chapter 5.

Chapter 2 Background

In this chapter, we first introduce peer-to-peer network including two major overlay structures, applications and block selection algorithm, understanding two important factors which affect the download time of the chunks, and then we introduce some bandwidth estimation methods.

2.1 Peer-to-Peer Network

2.1.1 Introduction

In the traditional client-server architecture, services must be provisioned by specific machine which has sufficient resource and is connected to the network. The machine is called “server”, which provides contents such as texts, multimedia streams or images, and “client” that connects to the server to get the data.

In the client-server communication model, a server could be a bottleneck of the service operation when the user grows up because of the restriction of resources such as computing power, network bandwidth, and the storage size of the server. Even worse, all services will be terminated when the server crashes. To reduce the impact of the server failure problem, server cluster or server farm has been used to avoid the single point failure problem. However, as the number of clients grows up, the cost of servers in such system would be too high to be affordable.

To enable the application to support a large number of clients, P2P architecture is employed in recent years. The end hosts participating in the service contribute their resources to help forward and replicate received stream to other clients. The clients, considered as peers in the system, are no longer only data receivers, but also data providers to increase system

capacity and service quality.

2.1.2 P2P file sharing applications

A great number of P2P file sharing applications have been developed over the world with different philosophies and operation modes after Napster raised the interest of P2P in late 1990s. Every peer has the ability to be the server. There is no restriction on what contents a peer can publish and what contents can be obtained from which peer. The most popular P2P file sharing applications are Gnutella, eDonkey and BitTorrent [7], etc.

a) Gnutella

Gnutella, developed by Justin Frankel and Tom Pepper of Nullsoft in 2000, is one of the earliest P2P file sharing tools. It is also one of those pure P2P applications that do not have a centralized server. Gnutella works as follows [8]. A Gnutella peer joins the network via at least one known peer, whose IP address is either obtained via a bootstrapping process or an existing list of pre-configured address. When the user wants to do a search, the source peer will send the search request to all actively connected peers as shown in Figure 2.1. The recipient peer answers the query if it has the data, otherwise it forwards the request.

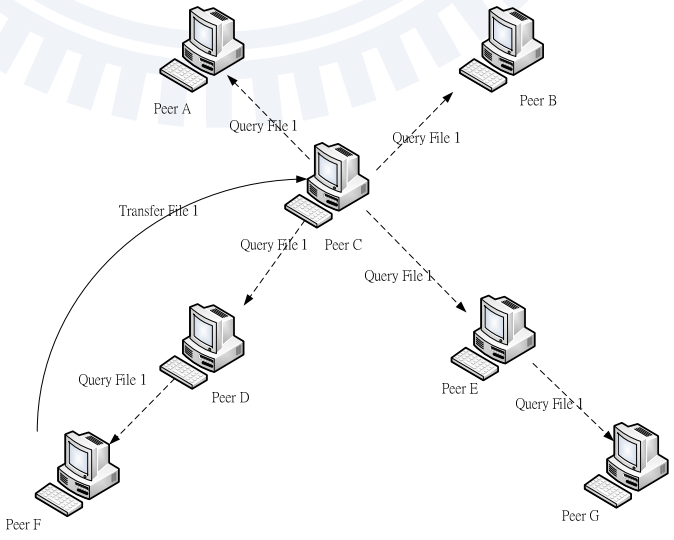

Figure 2.1 Purely decentralized P2P architecture

A noticeable feature is that Gnutella is under nobody's specific control and is effectively impossible to shutdown. However, because of too many search requests, when the number of Gnutella users grows too large, the search requests overwhelm the Internet and thus performance gets prematurely dropped.

b) eDonkey

The eDonkey was conceived in Sept. 2000 by MetaMachine Inc. The eDonkey network works as a hybrid P2P network which consists of clients and servers. The eDonkey client downloads and shares the files and the eDonkey server acts as a communication hub and distributes addresses of other eDonkey servers to the clients. The eDonkey network is based upon an open protocol. As a result, there are many versions of eDonkey clients and servers.

c) BitTorrent

BitTorrent (BT) was created in 2002 by Rram Cohen. It runs on an open protocol. To share a file or a set of files through BitTorrent, a torrent file is first created. The torrent file contains the metadata of the shared content, which includes the information of the tracker that coordinates the file distributed. When a BitTorrent user wants to retrieve a data, it first needs the torrent file. The client then contacts the "tracker server" listed in the torrent file which can be a single computer or a distributed set of computers. The "Tracker Server" in BT system logs who are downloading the file at the same time and helps peers to find each other.

The most significant contribution of BT is dividing a file into several fixed size pieces, called chunks. A peer can download a file from different peers for different chunks simultaneously. This idea greatly improves download efficiency and reduces a great deal of download time.

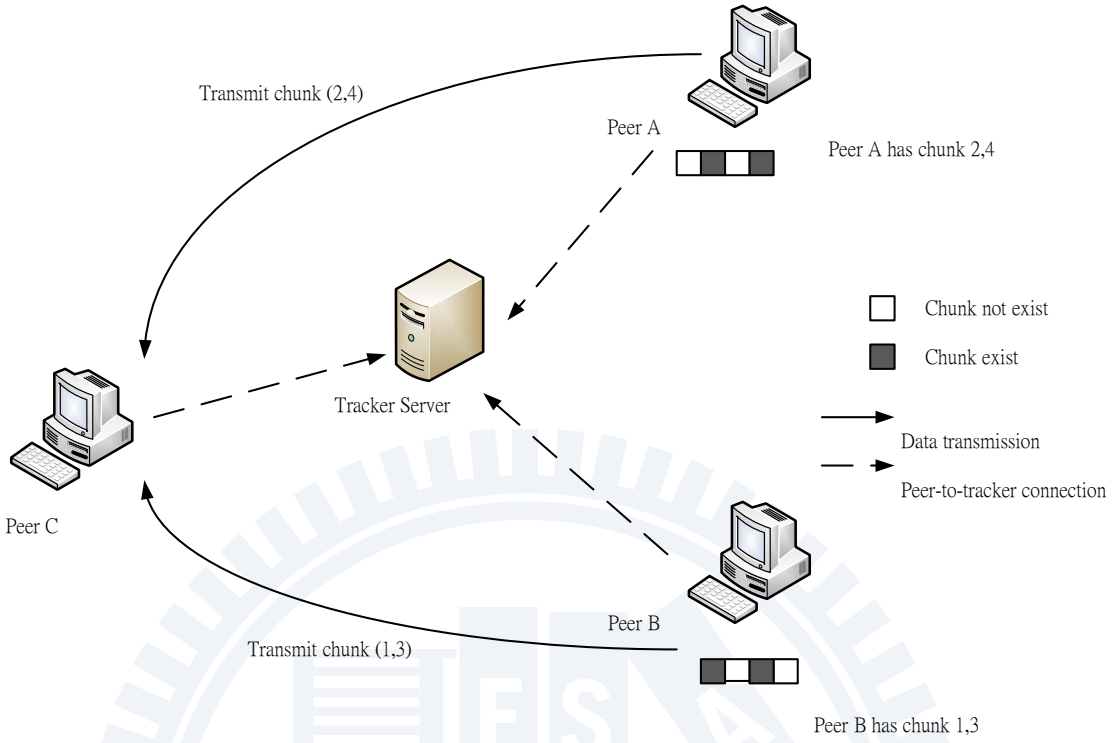

Figure 2.2 BitTorrent architecture

As Figure 2.2 shows, every peer connects to tracker server. When peer C wants to download the file, it asks tracker and gets a response which indicates that peer A has chunk 2 and chunk 4, peer B has chunk 1 and chunk 3. With the help of the tracker, peer C can download different chunks from different peers simultaneously.

The concept of "chunk" is used for many areas, such as P2P streaming. We will discuss P2P streaming in more detail in Section 2.2.4.

2.1.3 P2P VoIP-skype

Skype is a P2P VoIP (voice over Internet protocol) application. Compared with public VoIP standard such as SIP and H.263, Skype uses a proprietary protocol and relies on the Skype P2P network for user directory. As a result, Skype can easily scale to a very large size. Skype uses a two tier infrastructure, which consists of super-nodes and clients. When a peer

joins the network, it always starts as a client node. If a client has sufficient ability, Skype automatically promotes the client to a super-node. The Skype client finds at least one super-node as its default gateway, and first tries to establish UDP connection to the super-node. If that fails, it tries to establish a TCP connection on an arbitrary port to the super-node. The overlay used in Skype is shown in Figure 2.3.

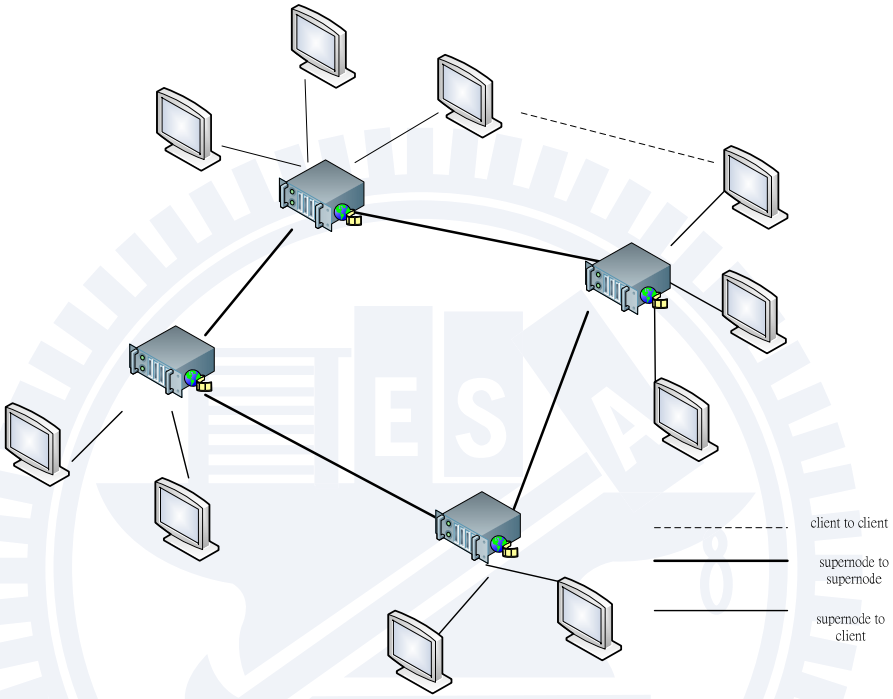

Figure 2.3 Two tier P2P overlay used in Skype

2.1.4 P2P streaming

To provide IPTV service in the Internet, the most intuitive approach is through centralized server. However as described above, client-server model has its restrictions. To solve this problem, researchers try to make IPTV based on P2P architecture. We take a look on the architecture in following.

a) Centralized server architecture:

IPTV based on centralized architecture is very similar to the traditional "client-server". The difference is that the server provides video contents instead of

images and text. Since IPTV has the potential to overwhelm the Internet backbone and heavy load on the server, we need to consider other approaches to overcome this problem.

b) IP layer multicast:

The first solution is to employ IP multicast scheme on routers. Instead of delivering to every user a copy of the content, server only sends video content to a multicast address. The video content will be replicated by the router so as to reduce the workload on the server. However, the routers in core network belong to different ISPs, a large portion of them do not enable router multicast function.

c) P2P architecture:

In P2P architecture, every peer downloads and shares video contents with each other. Because every peer plays as not only a client but also a server, thus, it reduces the workload on the server. The basic concept is that server provides content to a small subset of peers, and then shares the content with each other, as shown in Figure 2.4.

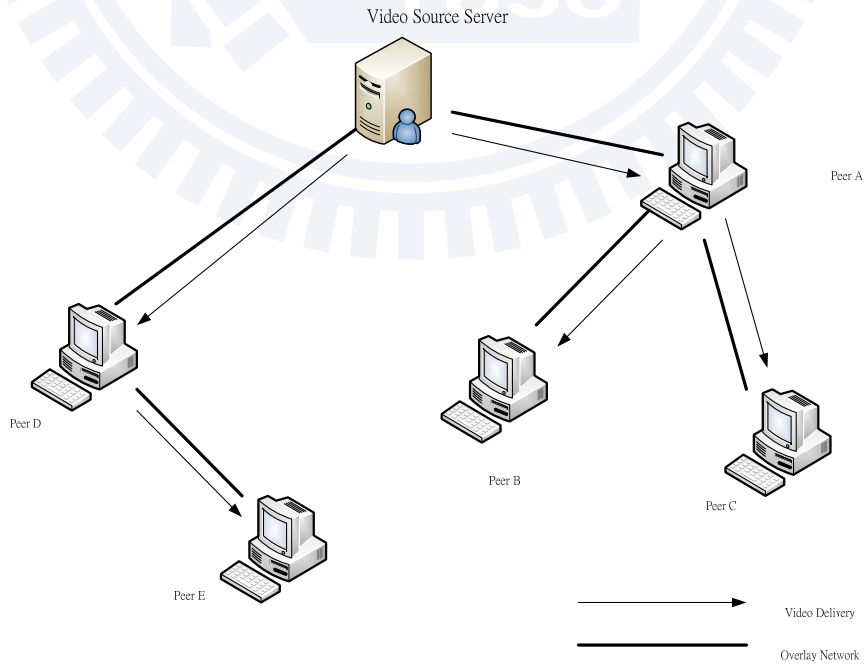

Figure 2.4 P2P architecture on video stream delivery

2.2 Overlay Construction

2.2.1 Tree-based overlay

The tree-based P2P streaming approach expands on the idea of the end-system multicast [9]. In the tree-based approach, an overlay construction mechanism organizes participating peers into multiple trees. For example, SplitStream, as shown in Figure 2.5, the original content is split into two stripes, and an independent multicast tree is constructed for each stripe such that a peer is an interior node in one tree and a leaf in the other. Each peer chooses the proper number of trees to join. To minimize the effect of churn and efficiently utilize available bandwidth in the network, participating peers are organized into multiple diverse trees. The content delivery is push mechanism where the branch node received media stream from their parents, and they also deliver the stream to their children nodes immediately. This is an efficient structure in terms of bandwidth and delay optimization. Therefore, the main component of the tree-based P2P streaming approach is the tree construction algorithm.

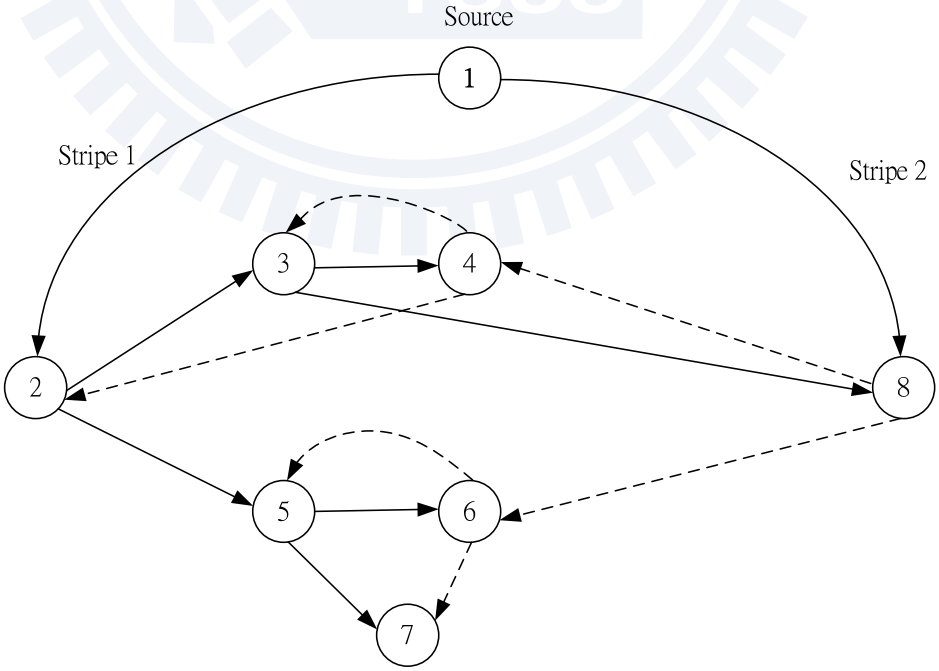

Figure 2.5 SplitStream multicast tree

The goal of the tree construction is to maintain multiple balanced, stable and short trees. A tree can always accept a new node. However, in the presence of churn, it could lead to the termination of services for all downstream sub-trees when a peer near to the root is crashed.

2.2.2 Mesh-based overlay

The mesh-based approach that employs the swarming content delivery is very similar to BitTorrent. The main advantage is its ability to efficiently utilize the outgoing bandwidth of participating peers as the number of the peer grows. In the mesh-based approach, peers form a randomly connected overlay. Each peer tries to maintain a certain number of partners which share the data with each other. Upon a new peer joining, it contacts a bootstrapping node to receive a set of peers that can be its partners. The new peer establishes the partner relationship with a subset of peers in PeerList, and uses a chunk scheduling algorithm to request the chunks from its partners.

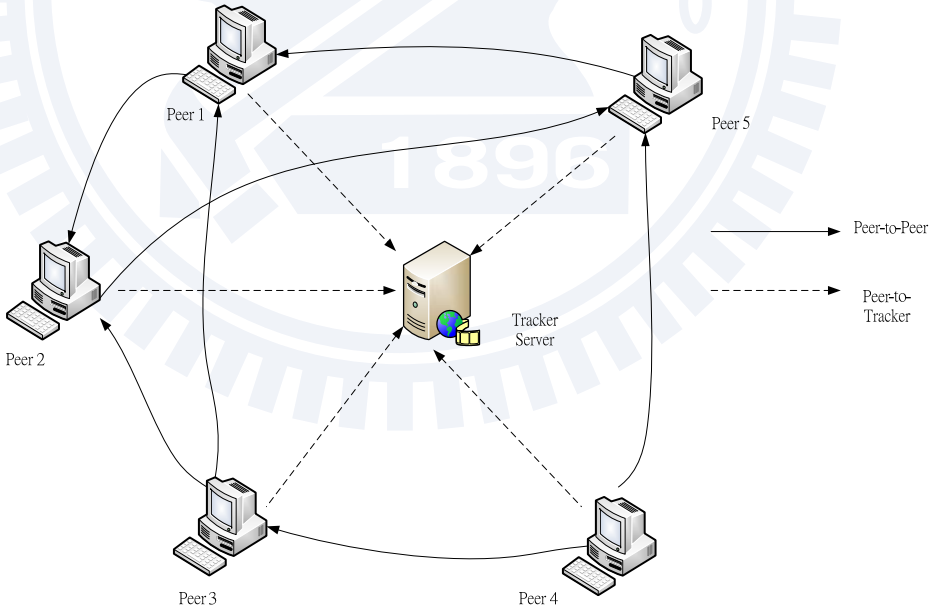

Figure 2.6 P2P Architecture: Mesh-based Overlay

Figure 2.6 shows the mesh-based overlay. The robustness and ease for implementation would be the biggest advantages of mesh-based overlay. Unlike tree-based overlay,

mesh-based overlay can easily adapt to the churn of the peer join and leave, so that it does not need any algorithm for repairing or maintenance.

2.2.3 Comparison of mesh and tree overlay

In this subsection, we describe the similarities and differences between two approaches.

Similarities: The tree-based and mesh-based approaches have a great deal of similarities as follows: First, in both approaches, they all enable peers to receive different chunk of the content. Each peer receives content from multiple parents and sends the content to multiple child peers. Second, both approaches require every peer to maintain a loosely synchronized playback time that is sufficiently behind source's playback time.

Differences: The key difference between mesh-based and the tree-based approaches are how the delivery tree of individual packet is formed [10]. In tree-based approach, each packet in the delivery tree is indeed pinned down by the tree construction mechanism because of the static mapping of descriptions to trees. In contrast, in the mesh-based approach, the delivery tree for individual packets is dynamically shaped as the packet traverses through the overlay. Thus, it can effectively utilize the available bandwidth than tree-based overlay.

2.3 Peer and block selection

Depending on whether the sender or the receiver takes the initiative in requesting the chunk, three modes may be used in the P2P delivery. In the pull mode, the receiver takes the initiative and pulls the chunks it wants from his partner. In the push mode, the sender takes the initiative, and pushes the data to the selected partner.

The tree-based P2P delivery usually uses the push mode. The peer simply pushes its chunk to downstream peers in the tree. The mesh-based P2P delivery is more flexible then tree-based, it my use either the push, the pull, or combined mode. Usually P2P streaming uses the pull-mode. The peers exchange the buffer map which indicates whether a peer has the

chunk or not with its partner, and based on it, peer makes a decision on which block to retrieve.

In either the push or the pull mode, the peer needs to decide upon a delivery strategy, in other word, which block to push/pull, and from which peer that the block is pushed and pulled. We call this block selection and the peer selection, respectively.

There have been some research efforts toward P2P block selection. Until now, some schemes such as sequential, random, rarest-first and round-robin, have been proposed and adopted in many existing media streaming system, we give brief description about schemes at the following.

a) Sequential: Choose the first block that can be pushed or pulled.
b) Random: Random scheme is a very simple scheme adopted by Chainsaw [11], it just randomly chooses the block.
c) Rarest-first: It is a heuristic algorithm with fast response time, and was proposed in CoolStreaming [12]. The main idea is choosing the block that is the rarest in the local neighborhood of the peers.

2.4 Bandwidth Estimation Techniques

In the context of data network, bandwidth quantifies the data rate at which a network link or a network path can transfer. For many data-intensive applications such as multimedia streaming or file transfers, the bandwidth available to the application directly impacts application performance. Bandwidth is also a key factor in several network technologies. For example, peer-to-peer network form their dynamic partner relationship based on available bandwidth between peers. However, IP network do not provide explicit feedback to end hosts, so hosts use active end-to-end measurements in order to estimate the bandwidth characteristics of paths they use [13].

There has been much work on developing techniques for estimating the capacity and

available bandwidth of network paths. Capacity is defined as the bandwidth of the narrow link (i.e., the link with the smallest bandwidth) on a path. Another important metric is the available bandwidth of a link. It is defined as a link related to the unused capacity of the link during a certain time period. The hop with the minimum available bandwidth is called the tight link of the end-to-end path.

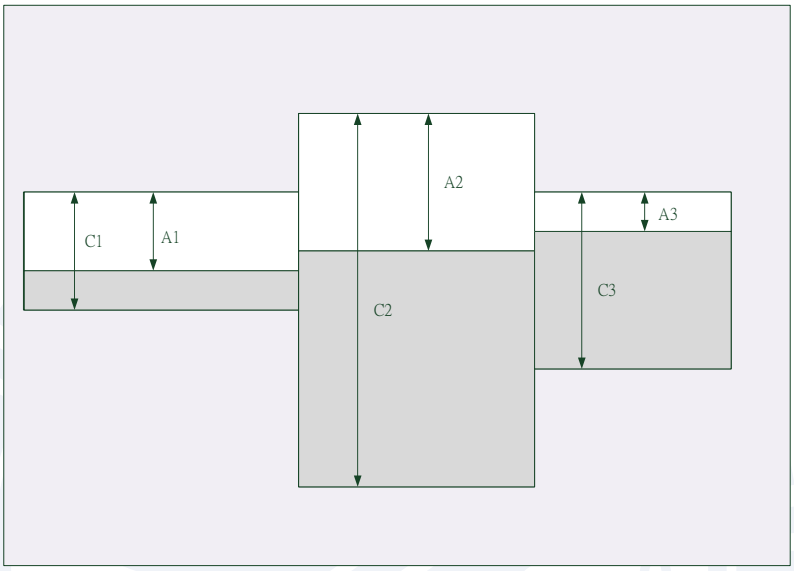

Figure 2.7 A pipe model for a three-hop network path.

Figure 2.7 shows a pipe model with fluid traffic representation of a network path, where each link is represented by a pipe. The width of each pipe corresponds to the relative capacity of the corresponding link. The white area shows the spare capacity, while the shaded area of each pipe shows the utilized part of that link's capacity. The minimum link capacity C1 in Figure 2.7 determines the end-to-end capacity, while the minimum available bandwidth A3 determines the end-to-end available bandwidth. As shown in Figure 2.7, the narrow link of a path may not be the same as the tight link.

We survey many tools and techniques that have been proposed for estimating the capacity and the available bandwidth of network links and paths.

Many of the proposed capacity estimation schemes are based on the packet pair/train dispersion probing principle. The source sends multiple packet pairs to the receiver. Each

packet pair consists of two packets of the same size sent back to back. The dispersion of a packet pair at a specific link of the path is the time distance between the last bits of each packet. Then receiver can estimate the path capacity from packet size and dispersion.

Turning to available bandwidth estimation, many of the recently proposed techniques will fall into two categories: packet rate method (PRM) and packet gap method (PGM) [14]. PRM-based tools are based on the observation that a train of probe packets sent at a rate lower than the available bandwidth would be received at the sending rate. However, the received rate would be lower than the sending rate, and the probe packets would tend to queue up behind each other, resulting in an increasing one-way delay trend, if the sending rate exceeds the available bandwidth. Available bandwidth can be estimated by observing the sending rate at which a transition between the two modes occurs.

PGM-based tools send pairs of equal-sized probe packets, spaced apart according to the transmission time of the probes on the bottleneck link. Gunawardena et al [15] have proposed an alternative approach to available bandwidth estimation based on measuring the RTT of probe packets.

Table 2.1 gives the names of these tools together with the target bandwidth metric they try to estimate and the basic methodology used.

Table 2.1 Taxonomy of popular estimation tools.

Tool	Author	Measurement metric	Methodology
Netperf	NLANR	TCP throughput	Parallel TCP connections
Iperf	NLANR	TCP throughput	Parallel TCP connections
ttcp	Muuss	TCP throughput	TCP connection
cap	Allman	Bulk transfer capacity	Standardized TCP throughput
treno	Mathis	Bulk transfer capacity	Emulated TCP throughput
pathChirp	Ribeiro	End-to-end available bandwidth	Self-loading packet chirp

IGI	HU	End-to-end available bandwidth	Self-loading periodic streams
pathload	Jain-Dovrolis	End-to-end available bandwidth	Self-loading periodic streams
cprobe	Carter	End-to-end available bandwidth	Packet trains
sprobe	Saroiu	End-to-end capacity	Packet pairs
pathrate	Dovrolis-Prasad	End-to-end capacity	Packet pairs and trains
nettimer	Lai	End-to-end capacity	Packet pairs
bprobe	Carter	End-to-end capacity	Packet pairs
pchar	Mah	Per-hop capacity	Variable packet size
clink	Downey	Per-hop capacity	Variable packet size
pathchar	Jacobson	Per-hop capacity	Variable packet size

Chapter 3 Proposed Approaches

As mentioned in the previous chapter, there are two important components supporting streaming applications in P2P networks: overlay construction and data scheduling. Data scheduling can be further divided into two parts: block selection and source peer selection. A good source peer selection can dramatically accelerate the speed of chunk download, and minimize the consumption of networks resources. Our focus is on the design of a source peer selection algorithm that can utilize peer bandwidth efficiently so as to reduce the download time of the chunk and enhance the stability of the system.

3.1 Design Philosophy

To reduce the download time of the chunk, the most intuitive idea is downloading the chunk from the one which has the highest bandwidth and the smallest queue size. Therefore, we combined these two factors and calculate the score for every partner which has the chunk we want. Then choose the partner who has the highest score and request the chunk from it. We proposed a source peer selection scheme based on swarming mechanism.

3.2 Main Scheme

We claim that there are three phases in data scheduling algorithm. These are block selection, source selection and finally sending the request.

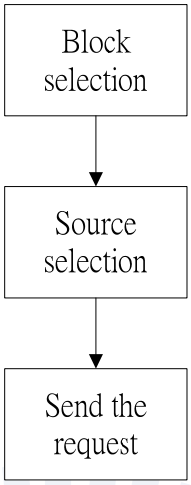

Figure 3.1 Data scheduling phases

If a user intended to get the chunk, his action involves the data scheduling. As shown in Figure 3.1, first, a user needs to decide which chunk to get. This action would be done by the block selection algorithm. Many researchers have proposed various methods, including randomly select, rarest first, sequentially select etc. When the peer decides which block to be requested, it will go to the next phases - source selection. Usually, it randomly chooses the partner who has the chunk, but it may not be an efficient method. Sometimes, it may choose the worst link, sometimes it may choose the good link, this will cause the download time not stable. Therefore it is hard to estimate how much data we should buffered. So, we propose an efficient source peer selection method to avoid the situation mentioned above. In the following, we describe our proposed approaches in more detail.

A peer who has the largest bandwidth and smallest loading can transfer the chunk more quickly. In this section, we will describe how to determine the best peer in order to request chunk from it and why we choose this factor.

Table 3.1 Notations list

Notation	Description	Formula
Bandwidth[k]	Bandwidth of the partner k	
Lastsent[k]	Data sent by the partner k in the last five second	
Queuesize[k]	Queue size of the partner k	
Score[k]	Score of the partner k	Score[k]=α*bandwidth[k]+β*(lastsent[k]/ queuesize[k])

Table 3.1 shows the notations used in our proposed scheme. Suppose the peer p_d request the chunk V and there are M partners $p_1, p_2 \ldots\ldots p_m$ that have the chunk V .

1) Bandwidth: bandwidth quantifies the data rate at which a network link or a network path can transfer. Bandwidth directly impact the download time of the data. With the large bandwidth, we can transfer the data much quickly so that we choose the bandwidth as the first factor in our proposal.
2) Last-sent: last-sent indicates the partner who had sent the packet at the last five seconds, it means this partner is active in the last five second, so if two partners have the same bandwidth, we choose the partner who has sent more packet within the last five second. The value 5 is adjustable depending on the system need.
3) Queue-size: queue size means how many packets are waiting in the queue. If there are too many packets in the queue, it means this partner has too heavy loading so that it can't handle the request.

We use these three factors to calculate the score for every partner k. α and β are used to normalize the factors and represent their relative importance. To achieve our goal, every peer will record the data it sent and the message queued in the buffer.

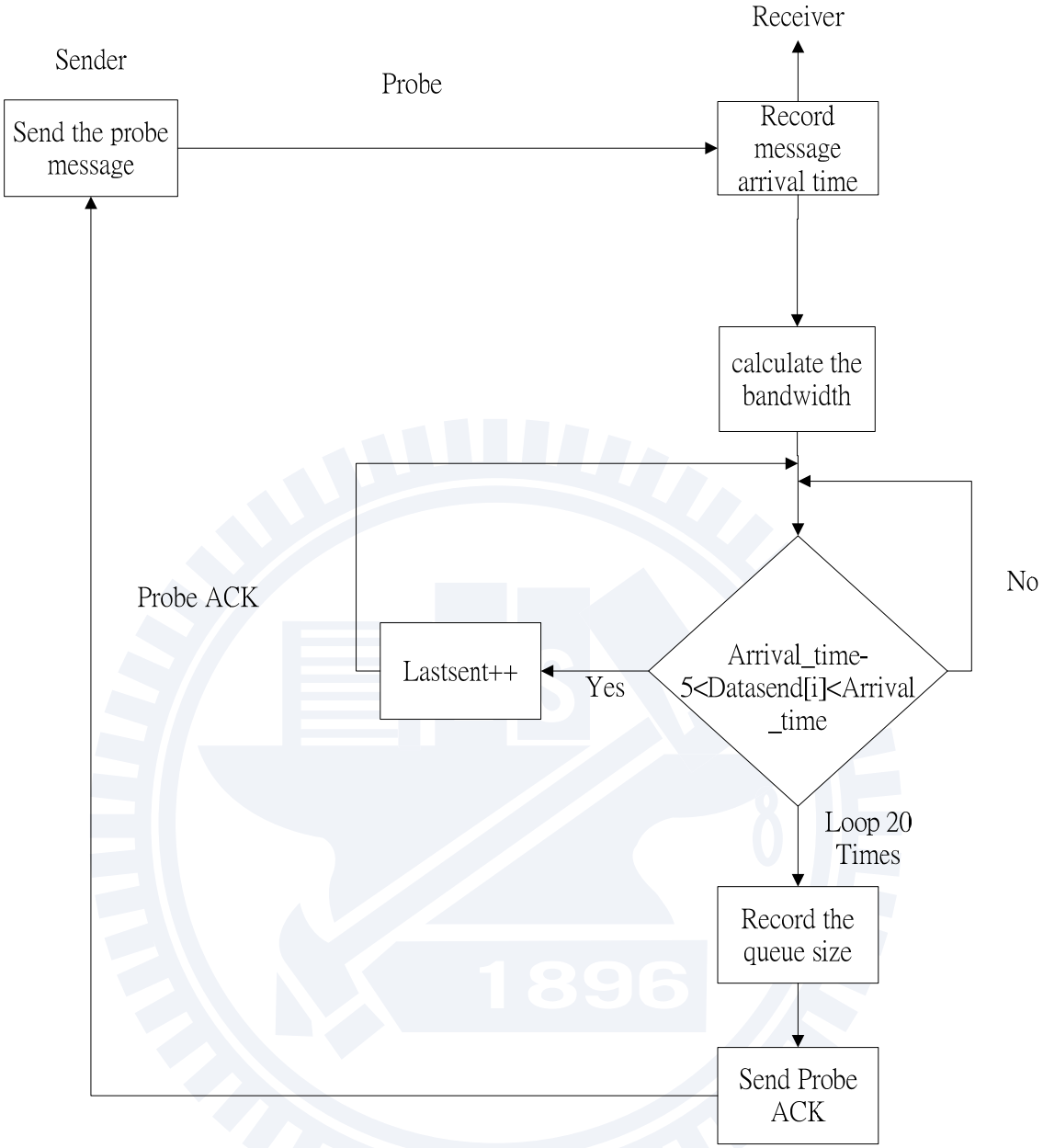

Figure 3.2 probe flow chart

Figure 3.2 shows how to obtain the three factors we want through the probe message. A sender will send the probe message to every partner. When a partner receives the probe message, it records the arrival time of the probe message first. Then the receiver uses the arrival time and the packet size of the message to estimate the link bandwidth. We use one-way delay to estimate the link bandwidth instead of packet-pair because of too heavy overhead in packet-pair method. Second, since every peer would record the data sending time, we can compare the arrival time of probe message subtracted by 5 and determine whether this

partner have sent the packet at the last five second or not. If yes, last-sent plus one, this loop will run twenty times and then proceed to next step. Third, extract the queue size from the queue and send the probe ACK message which contains the information about the bandwidth of the link, last-sent and queue-size of the partner.

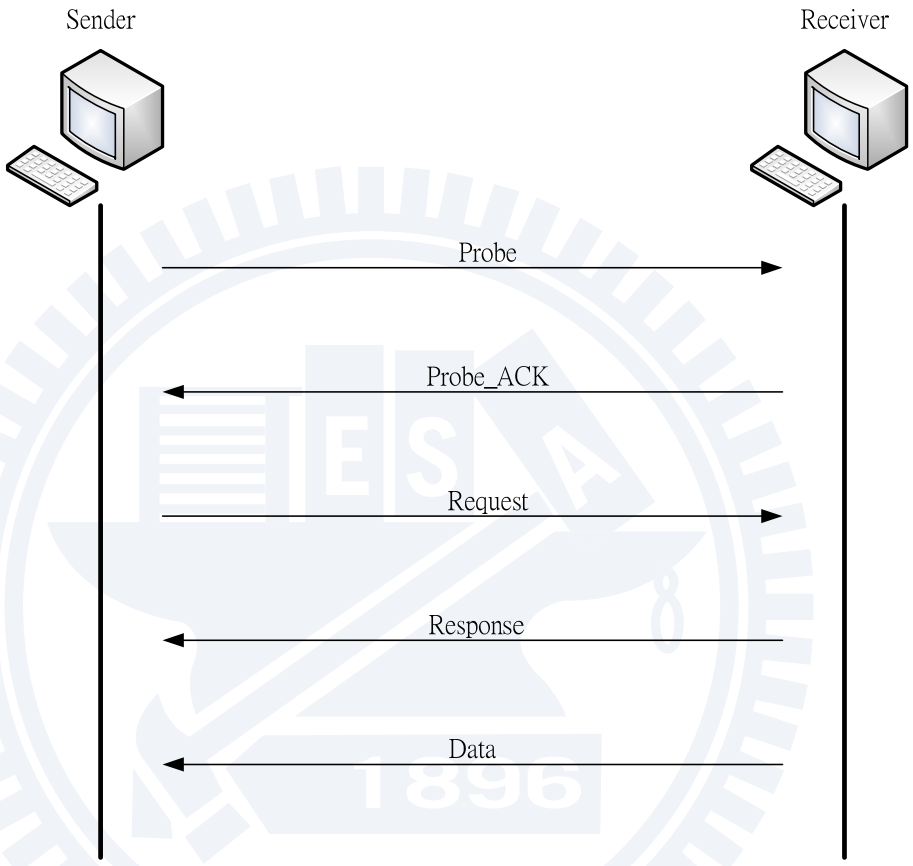

Figure 3.3 Message flow for a peer to retrieve data

Figure 3.3 shows the message flow of a peer who wants to get the data. As described above, every partner send the probe ACK message to the sender containing the information about all of the factors used to calculate the score. When the sender receives all partners' ACKs, it will calculate the score for every partner based on the information the ACK message back. The sender will choose the partner who has the highest score and send the request to it and get this chunk. This source peer selection algorithm would be involved every time when one peer wants to get the chunk.

1. For each peer p_s in sourcePeerSet
2. Calculate score p_d for every peer in the sourcePeerSet
3. Choose the peer that has the highest score
4. Send the request to this peer and get the data from it

Figure 3.4 The pseudo code for the proposed source peer selection algorithm

3.3 Summary

With the help of probe message, source peer selection algorithm allows the peer to select the best peer of its partner instead of making random selection. Our proposed schemes feature the following benefits.

1. Reducing the download time.
2. Efficiently use the link bandwidth.
3. Enhance the system stability
4. Reduce the playback buffer size.

Chapter 4 Simulation Evaluation

In this chapter, simulation is performed and numerical result shows the improvement between our proposed source peer selection and the random source peer selection. We introduce the simulation environment and tool first. Then we present the simulation results of random source peer selection and proposed source peer selection. We compare the download time of the chunk and overhead of the two source peer selection.

4.1 Simulation environment

We evaluate our proposed algorithm on the OMNet++ [16] simulator. OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, with an Eclipse-based IDE and a graphical runtime environment. OMNeT++ provides the basic machinery and tools to write simulations, but it itself does not provide any components specifically for computer network simulations, queuing network simulations, system architecture simulations or any other area. Instead, these application areas are supported by various simulation models and frameworks such as the Mobility Framework or the INET Framework. These models are developed completely independent of OMNeT++. This enables the software to be better tested and functions developed independent of each other.

Our simulation will only provide the Application Layer functionality, as defined in the OSI 7 Layer Model. The simulation will not include Transport (TCP) or Internetwork (IP) layers, as these are provided by other models. Figure 4.1 shows the developing flow of how to develop our simulation with the OMNet++.

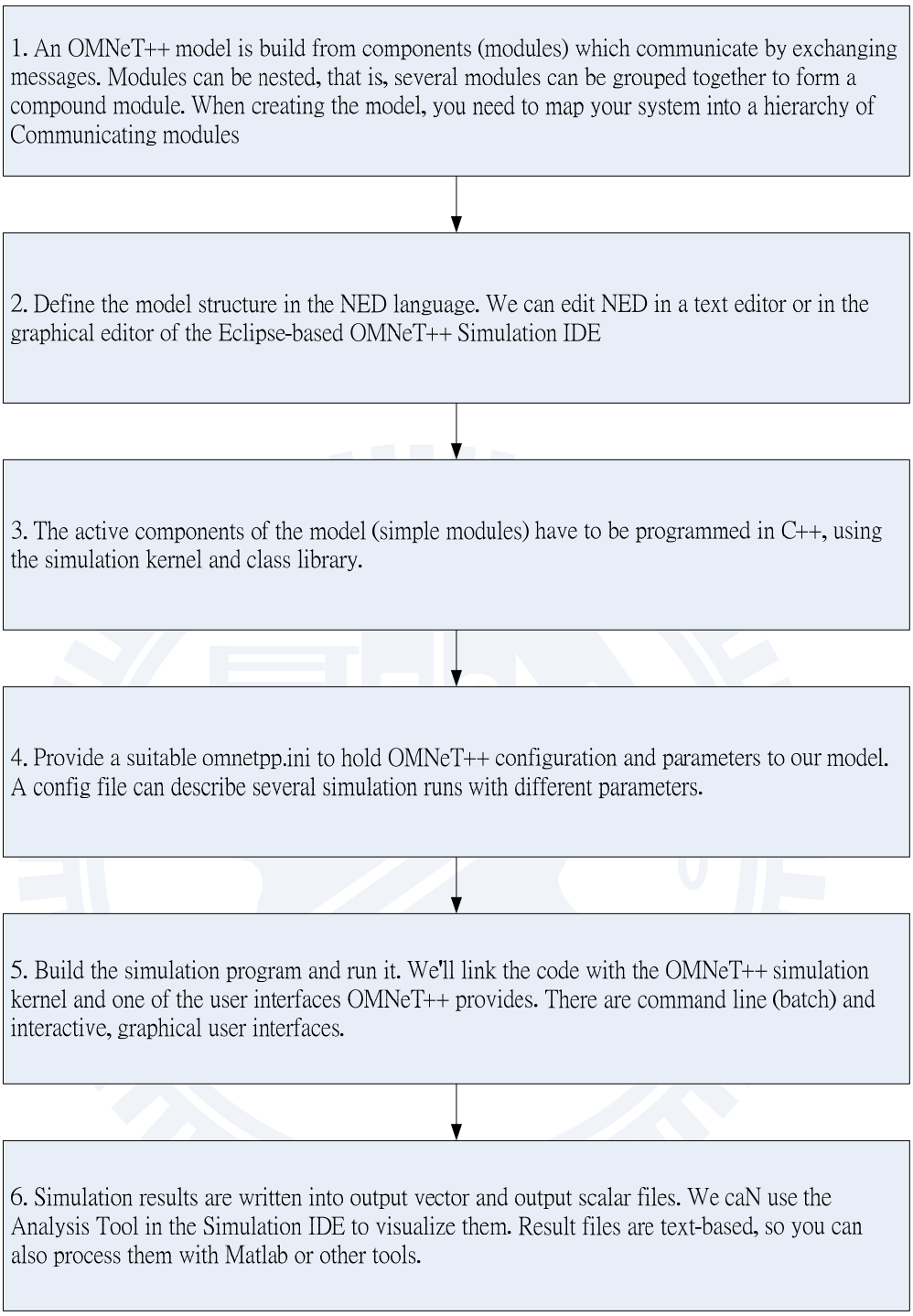

Figure 4.1 Developing the simulation with OMNet++

4.2 Simulation architecture and setting

In this section, we describe our network architecture and the parameters setting.

The system architecture covers the high level layout of the simulation network. For our

simulation a simple star network will be used, with each peer connected to a central switch, wand a dedicated connection to each node. This architecture is simple enough to enable the simulation to be scaled to many hundreds of nodes. Variations in bandwidth between nodes can be done via OMNet++, which provides the functionality to simulate delay and bandwidth constraints on each connection. This model can be easily extended to a more distributed architecture, with a number of switches or routers. Figure 4.2 shows the architecture of the basic simulation network.

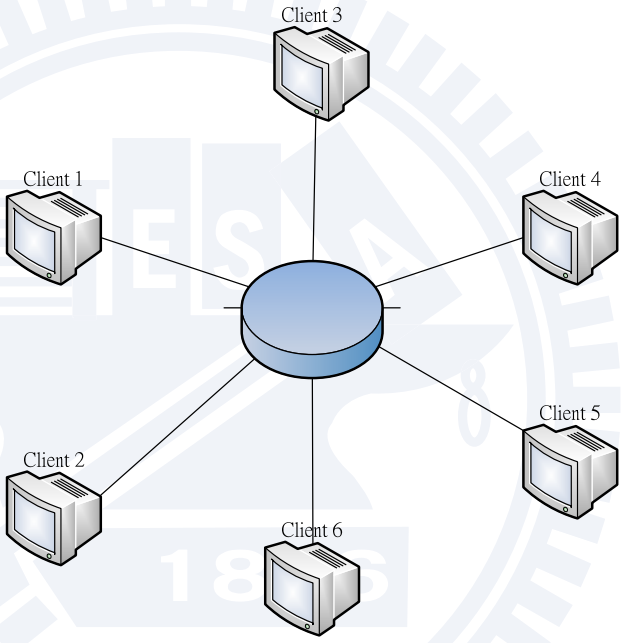

Figure 4.2 Simulation network architecture

To implement the architecture two modules were built, a peer module and a switch module. The implementation of the system network architecture in the OMNet++ GUI environment (TKenv) is shown in Figure 4.3.

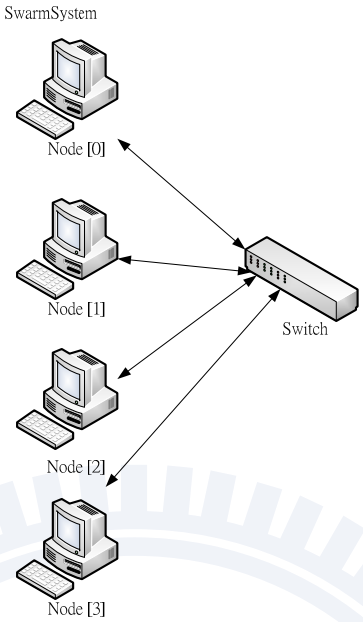

Figure 4.3 System architecture as implemented in the OMNet++ GUI

“Sim-time” indicates how long we run the simulation program. “Run” indicates how many times we run the program. “Sendperiod” means the period we send the packet. And every link has different bandwidth. All the parameter settings are showed in the Table 4.1.

Table 4.1 Parameters setting in the simulation environment

Parameter	**Value**
Sim-time	5min
Run	10
Data Packetsize	62500bytes
Probe Packetsize	125bytes
Sendperiod	1s
Link[k]	256Kbps+k*256Kbps

4.3 Simulation result

In this simulation, we compare our proposed peer selection with random peer selection. We show the improvement of the download time.

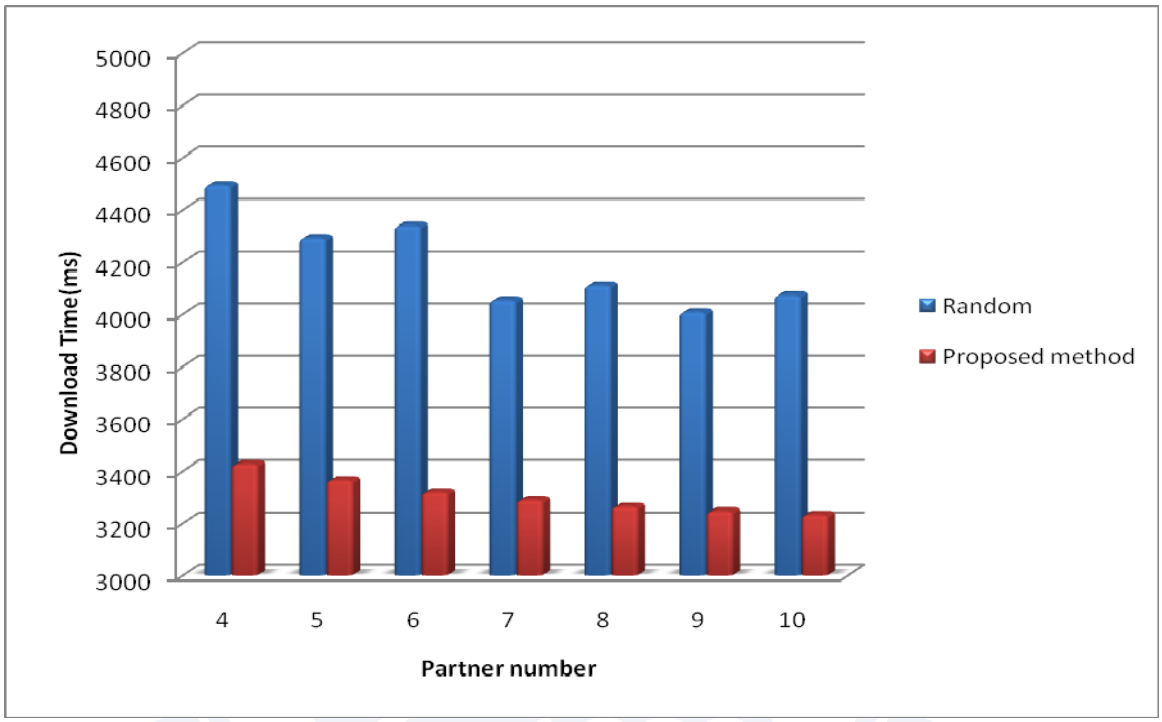

Figure 4.4 Simulation result: download time of two method

Table 4.2 Simulation result: improvement of the download time

partner	4	5	6	7	8	9	10
random	4495.2	4291.2	4339.6	4054.7	4110.3	4007.7	4073.4
proposed	3427	3364	3320	3289	3264	3247	3233
improve	23.8%	21.6%	23.5%	19.0%	20.6%	19.0%	20.6%

Unit: ms

Figure 4.4 shows the bar chart of two methods and Table 4.2 shows the percentage of the downloading time reduction. As we can see, our proposed method dramatically reduces the download time of the chunk. As the partner grows, average download time of the data decrease because of larger link bandwidth has been added. However, random peer selection

randomly select the source peer, it may sometimes choose the peer who has a large bandwidth, sometimes choose the peer who has a bad link bandwidth, thus its average download time of the data cannot always decrease as the number of partners grows up.

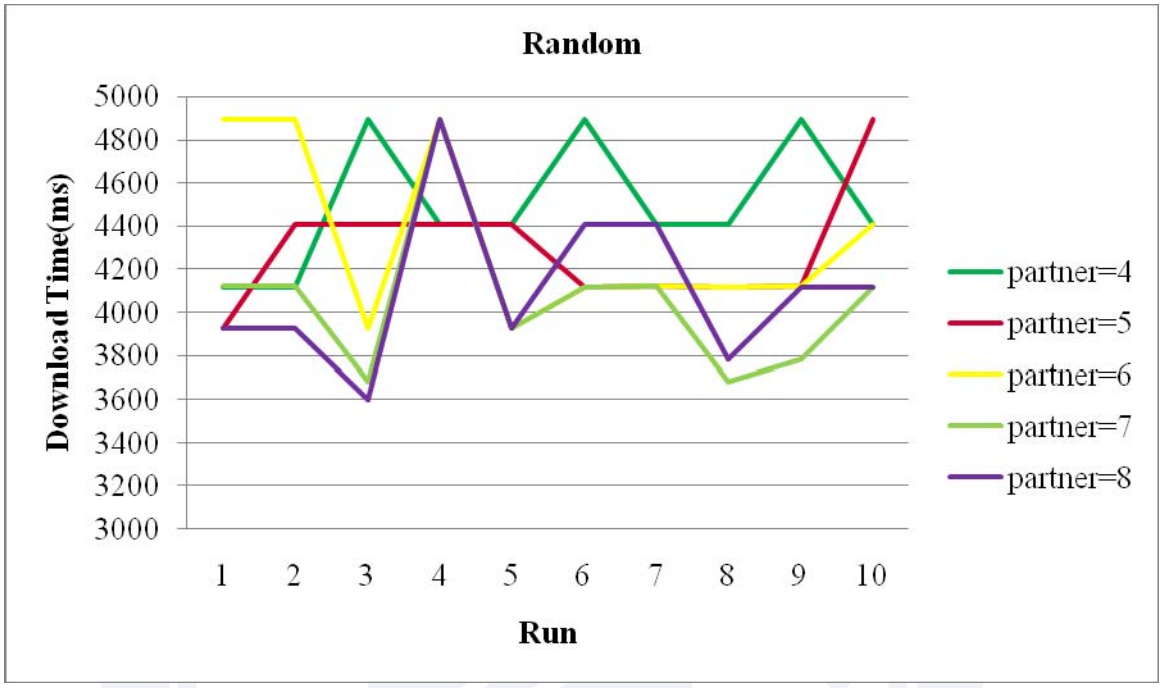

Figure 4.5 Line chart of the random peer selection

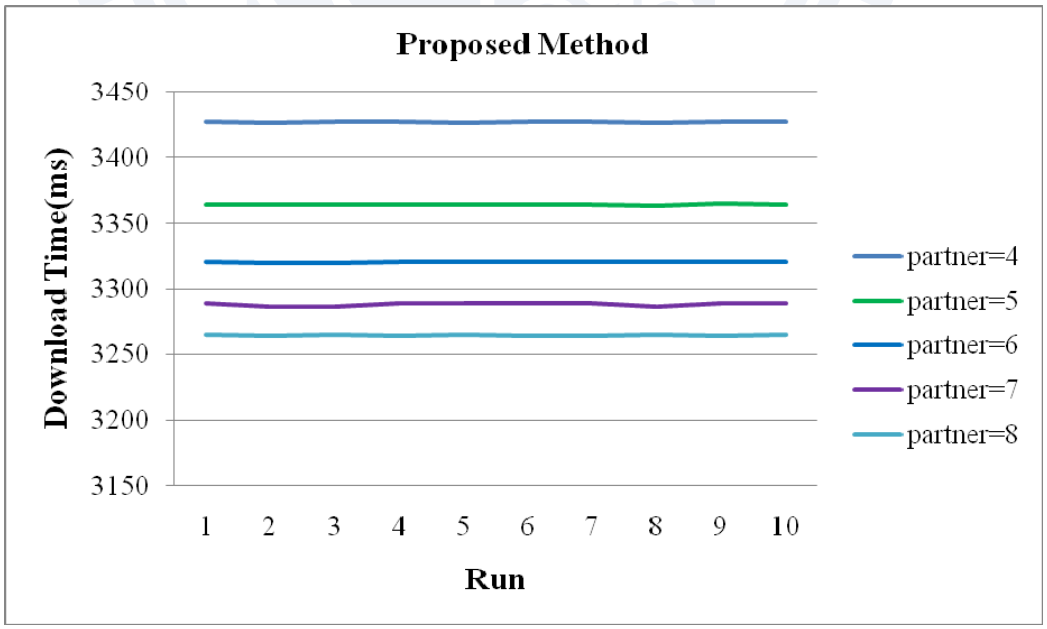

Figure 4.6 Line chart of proposed peer selection

Figure 4.5 and 4.6 shows download time of every run. As we can see, in random peer selection every run has the different download time, sometimes are fast and sometimes are slow. Its fluctuation are very dramatic, that results in difficult buffer design because it is hard to estimate the download time so we don't know how much of the data should be buffered. Our proposed peer selection in every run has almost the same download time when the partner's conditions are the same. So it can increase the stability of the system.

Table 4.3: Increased overhead (Kbytes) of proposed peer selection

Partner	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Overhead	34.25	43.45	52.5	61.25	71	79.875	79.875
Download	4187.5	4250	4312.5	4343.75	4343.75	4375	4375
Percentage	0.82%	1.02%	1.22%	1.41%	1.63%	1.83%	2.04%

Unit: Kbytes

As the number of partners grow, the overhead (Kbytes) increase too. But even with ten partners, overhead (Kbytes) just increases 2.04%, we think it is acceptable.

4.4 Summary

In Chapter 4, the comparison of the random peer selection and our proposed peer selection is presented and the simulation results show the improvement on packet download time and stability of the system.

To reduce the download time, how to choose the best source peer is what our proposed scheme addresses, and simulation results of the download time demonstrate our improved approach. Smaller download time and system stability is the major improvement of our proposed scheme.

Chapter 5 Conclusion

P2P Streaming system becomes more and more popular in recent years. Many researchers focus on overlay construction and block selection. Only few of them focus on source peer selection, and usually use the random source peer selection, so it cannot efficiently utilize the link bandwidth and increase the complexity of buffer design because of fluctuation of the packet download time.

In this thesis, we proposed an efficient source peer selection into P2P system. It focuses on improving the packet download time and system stability. The source peer with large bandwidth sent the most of the packets at the last five second and smaller queue size would be chose by the peer first. This can efficiently utilize the best partner of peers and thus reduce the packet download time.

To evaluate the efficiency of our proposed scheme, we establish a simulation environment implemented in OMNet++ to simulate our proposed source peer selection. The result shows that our proposed scheme's packet download time is much smaller than the random source peer selection. It can reduce about 20-24% packet download time with just a little overhead (Kbytes). And the simulation also shows that our proposed scheme is more stable than random peer selection. The stable system can also reduce the complexity of the buffer design.

For further simulations and more accurate simulation results we should implement out proposed source peer selection in practice. Also we should add more simulation parameters to enhance the reality of the simulation result of our proposed source peer selection.

Reference

[1] Napster, http://free.napster.com/.

[2] BitTorrent, http://www.bittorrent.com/.

[3] Skype, http://skype.com/.

[4] PPLive, http://www.pplive.com/.

[5] PPStream, http://www.ppstream.com/

[6] Hei, X., C. Liang, et al."A measurement study of a large-scale P2P IPTV system." IEEE Transactions on Multimedia, 9(8), 1672-1687, Dec. 2007.

[7] Jin Li, "On peer-to-peer(P2P) content delivery", Peer-to-Peer Networking and Application, Volume 1, Issue 1, 45-63 Mar. 2008.

[8] The Gnutella protocol specification v4.0 http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[9] Y. Chu, S. G. Rao, and h. Zhang, "A case for end system multicsat", ACM SIGMETRICS, 2000.

[10] Magharei, N., Rejaie, R., Yang Guo, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches", INFOCOM 2007. 26th IEEE International Conference on Computer Communications. , 1424-1432, May. 2007.

[11] Pai, V., Kumar, K., et al., "Chainsaw:Eliminating trees from overlay multicast", In Proceedings of IPTPS'05, Ithaca, NY, USA, Feb. 2005.

[12] Zhang, X., Liu, J., Li, B., Yum, T.S.P., "CoolStreaming/DONet: A Data-Driven Overlay network for Efficient Live Media streaming", IEEE INFOCOM 3, 2102-2111, Mar. 2005

[13] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools", IEEE Network, Vol.17, NO.6, Nov.-Dec. 2003.

[14] Karthik Lakshminarayanan, et al., "Bandwidth Estimation in Broadband Access Networks", IMC'04, 314-321, Taormina, Sicily, Italy, Oct. 2004.

[15] D. Gunarwardena, et al., "Network: characteristics: Modelling, Measurements and Admission Control", IWQoS, 2003.

[16] http://www.omnetpp.org/.