Analysis of Streaming Media Systems

Analysis of Streaming Media Systems

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 21 oktober 2010 om 10:00 uur

door

Yue Lu

elektrotechnisch ingenieur
geboren te ChangSha, Hunan provincie, China.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. P.F.A. Van Mieghem

Samenstelling promotiecommissie:

Rector Magnificus,	Voorzitter
Prof.dr.ir. P.F.A. Van Mieghem,	Technische Universiteit Delft, promotor
Dr.ir. F.A. Kuipers,	Technische Universiteit Delft, co-promotor
Prof.dr.ir. R.E. Kooij,	Technische Universiteit Delft
Prof.dr.ir. E.R. Fledderus,	Technische Universiteit Eindhoven
Prof.dr. T. Plagemann,	University of Oslo, Norway
Prof.dr. U.R. Krieger,	Otto-Friedrich University Bamberg, Germany
Dr.ir. D. De Vleeschauwer,	Alcatel-Lucent Bell NV, Belgium

Keywords: Multimedia Streaming, Server-Client and Peer-to-Peer, Quality of Experience, System Performance Analysis

Copyright © 2010 by Y. Lu

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the author or Delft University of Technology.

to Mom, Dad, and Jinglong

Contents

1 Introduction **1**
1.1 Peer-to-Peer (P2P) 1
1.2 Quality of Experience 3
1.3 Streaming Systems 4
1.3.1 Part I: TV Streaming Systems 4
1.3.2 Part II: Video-on-Demand Streaming Systems 8
1.3.3 Part III: Video Conferencing Systems 9

I TV Streaming Systems **13**

2 P2PTV: Measurement Study **15**
2.1 Introduction 15
2.2 Related work 16
2.3 How does SopCast work? 17
2.3.1 Experimental settings 17
2.3.2 Dissecting the SopCast protocols 18
2.4 Quality of Experience 23
2.4.1 PlanetLab Experiments Results 23
2.4.2 Subjective Measurements 33
2.4.3 Conclusions 34

3 P2PTV: Analytical model **37**
3.1 Related Work 37
3.2 Model with assumptions 37
3.3 Computation of the blocking probability 38
3.4 Case Study 40

4 IPTV: Analytical model **45**
4.1 Introduction 45
4.2 Related work 46

4.3 Model with assumptions 46
4.4 Computation of the blocking probability 46
4.5 Case Study 49
4.6 Blocking Comparison between IPTV and P2PTV 50

5 CDN: Analytical model 51
5.1 Introduction 51
5.2 The service architecture 52
5.3 Model with assumptions 53
5.4 Computation of the blocking probability 56
  5.4.1 Definition of blocking probability $B(k)$ 56
  5.4.2 Computation of $B_m(k)$ 57
  5.4.3 Computation of $B_{u;i}$ 58
5.5 Experiments 60
5.6 Summary and Conclusions 62

II Video-on-Demand (VoD) Streaming Systems 65

6 P2PVoD Model 67
6.1 Introduction 67
6.2 Related work 68
6.3 A general fluid model for P2PVoD 68
  6.3.1 Model description 70
  6.3.2 Start-up phase ($0 \leq t < \tau_i$) 72
  6.3.3 Seed Appearance phase ($\tau_i \leq t < L_i$) 72
  6.3.4 Seed Departure phase ($t \geq L_i$) 72
  6.3.5 Steady-state 73
6.4 Linearization of the P2PVoD model 75
6.5 Experiments 77
  6.5.1 General non-linear system 78
  6.5.2 Linearized system 80
6.6 Conclusion 82

7 P2PVoD system with helpers 83
7.1 Limitations of P2PVoD system without helpers 83
  7.1.1 Start-up phase problem 83
  7.1.2 Burden on peers 84
7.2 Solution: Adding helpers into the system 85
  7.2.1 Solving the Start-up phase problem when $v > bw_{up}$ 86
  7.2.2 Usefulness of helpers in Steady state 89

7.2.3 Case study of the system with helpers when $v > bw_{up}$ 90
7.3 Conclusions 92

III Video Conferencing (VC) Systems 93

8 VC Background 95
8.1 Classification of VC 95
8.1.1 Different terminals 95
8.1.2 User interface 95
8.1.3 Signaling protocol 96
8.1.4 Network structure 96
8.2 Available Desktop VC applications 97

9 Measurement Study of Multi-Party Video Conferencing 99
9.1 Introduction 99
9.2 Related work 100
9.3 Experiments Set-up 101
9.4 Measurement results 102
9.4.1 Login and Call establishment process 102
9.4.2 Packet size distribution and traffic load 103
9.4.3 Quality of Experience (QoE) 105
9.4.4 Worst-case study 109
9.5 Summary and Conclusions 109

IV Conclusions 111

10 Research context, methods, and contributions 113
10.1 IPTV 113
10.2 P2PTV 114
10.3 CDN 115
10.4 P2PVoD 116
10.5 VC 117

11 General Conclusion on “To Peer or Not to Peer” 119

12 Future Work 121

A Deductions 123
A.1 $B_{Engset}(i)$ in IPTV and $B_{Engset}(k)$ in CDN 123
A.2 How to set the value of v in the start-up phase, without helpers 124

A.3 Deduction of γ_i in the system with helpers 125

B Strategy and Protocols with helpers **127**

C Notations **129**

C.1 Notation P2PTV model 130

C.2 Notation IPTV model 131

C.3 Notation CDN model 132

C.4 Notation P2PVoD model 134

D Abbreviations **135**

Bibliography **137**

Samenvatting (Summary in Dutch) **143**

Acknowledgements **145**

Curriculum Vitae **147**

Summary

Multimedia services have been popping up at tremendous speed in recent years. A large number of these multimedia streaming systems are introduced to the consumer market. Internet Service Providers, Telecommunications Operators, Service/Content Providers, and end users are interested in the mechanisms they use, the Quality-of-Experience they provide (e.g. video/audio quality, audio-video synchronization, communication delay, start-up time, etc.), the resources they need, the system stability, and the service availability. The multimedia streaming systems analyzed in this thesis include IP-layer multicast TV (IPTV), Peer-to-Peer TV (P2PTV), Content Delivery Networking (CDN), Peer-to-Peer Video-on-Demand (P2PVoD), Server-to-Client Video Conferencing (IPVC) and Peer-to-Peer Video Conferencing (P2PVC). A summary of the work presented in this thesis can be found in Figure 1.

	E2E Blocking	QoE (view of users)	Network & Traffic	System Stability
IPTV	Analytical models & Measurements			
P2PTV		Measurements	Measurements	
IPTV+IPVoD (CDN)	Analytical models			
P2PVoD	Analytical model & Simulation			Analytical model & Simulation
IPVC		Measurements	Measurements	
P2PVC		Measurements	Measurements	

Figure 1: Overview of the research presented in this thesis.

This thesis aims to study various kinds of popular streaming systems, through analytical models, measurement experiments, and simulations, to reveal their characteristics and performance in different aspects. Based on this research, we can not only better understand the behavior and limitations of existing systems and find out the key parameters that affect their performance, but also investigate the potential problems and predict the system performance for future cases. By comparing the two general streaming content delivery methods (Server-Client and Peer-to-Peer), we gain in-depth insights on "which is better" and "what determines better" for different services and in different scenarios.

In Part I of this thesis, TV streaming systems will be analyzed. In Part II, Video-on-Demand streaming systems will be discussed. Video Conferencing systems will be investigated in Part III.

Chapter 1

Introduction

The importance of streaming content distribution (e.g. TV streaming, Video-on-Demand streaming, Audio/Video Conferencing, Gaming) is already evident and is only expected to grow in the future. Advances in content distribution are important for entertainment, social and environmental reasons: For instance, viewing TV programs and movies on Television or PC or Mobile Phones becomes one significant part of our daily lifes. Besides, interactivity is becoming more and more important. The uses of ICT and media applications also enable a more efficient use of energy, for instance Video Conferencing applications that provide an alternative to traveling to conferences/meetings.

Many more examples can be envisioned, all proving the value of streaming content distribution to our society. However, currently service providers do not know how to better distribute the streaming content to achieve the fastest and most efficient delivery while providing the best Quality of Experience to their customers. What kind of architecture they should implement (Server-Client or Peer-to-Peer) and what key technologies should be involved will therefore be discussed in this thesis. In this thesis, via analytical models and measurement studies, we will mainly present which challenges we need to face, and what key parameters or methods we should properly consider in order to realize a more stable and well-performed streaming system, for different kinds of streaming services.

In the following of this Chapter, the description of Peer-to-Peer technology, the defintion and the usage of Quality of Experience, and different kinds of streaming systems addressed in this thesis will be introduced.

1.1 Peer-to-Peer (P2P)

In the last years, the success of Peer-to-Peer (P2P) systems has grown exponentially thanks to the deployment of file-sharing applications. Following the achievement of P2P file sharing, nowadays the use of P2P technology to distribute media streaming content is receiving great interest from both commercial and academic research. This technology has enhanced the distribution of information on the Internet by enabling efficient cooperation among end users. Participants in these systems are viewed as logical and functional equals. This is in contrast to pure Server-Client protocols, where participants either provide resources or receive resources.

Based on the type of content distribution, a P2P system can be mainly classified into tree-based and mesh-based systems.

In tree-based systems peers are hierarchically organized in a tree structure where the root is the original content source provider. The content is spread as a continuous flow of information from the source down to the tree. The positions of each peer in the tree are fairly static. In single-tree systems all the traffic load is supported by the interior nodes of the tree, while the leafs are just receiving data. Thus, if an interior node does not have the required computational or bandwidth resources to serve all its children, the position of this interior peer could move down to a lower level of the hierarchical tree, otherwise its sub-tree nodes will suffer from high delays in data reception. If a leaf of the tree fails or leaves, the system will not suffer. On the contrary, if an interior node fails, peers on its sub-tree will lose data until the tree structure is repaired.

In mesh-based systems peers are not organized in a hierarchical structure. Here, the source splits the content in a series of chunks and distributes them to different peers. Peers establish relations with each other based on the information about what content chunks they have. A peer establishes relations with potential providers in order to get its missing chunks. A peer that cannot get data successfully from a failed node will download his missing chunk from another peer who contains the content he needs, without the need of repairing mechanisms. If several peers contain the same missing chunk, selecting one peer can depend on the resources they have at that moment (i.e., access capacity, stability, processing capability, the time they are already in the system, etc.). In these systems it is possible to introduce the concept of fairness (i.e., the “choke/unchoke” and “interested/not-interested” mechanisms try to ensure fairness among participating peers in BitTorrent[1]). It is also possible to evaluate the behavior of the other peers and to penalize or reward them accordingly (i.e., a policy known as “tit-for-tat”[2] and a mechanism called “optimistic unchoking”[3] have been implemented to encourage sharing in BitTorrent). Besides, peers can download the rarest chunks

---

[1] BitTorrent peers use two different states: “interested” and “choked” to maintain proper transfer. The “interested” state indicates that the peer is wishing to download specific chunks from other peers. The not-interested message indicates that the peer no longer needs any data. Similarly, a peer willing to serve any chunks sends “unchoke” messages to other peers. Once a peer decides to stop sharing, it transmits “choke” messages to its connected peers.

[2] Following the initial handshake, peers inform each other regarding availability of content. Because a serving peer can decide whether to allow uploading to a requesting peer, peers not sharing any content tend to be reciprocated in the similar way (therefore cannot download any content fast).

[3] The tit-for-tat mechanism is unfair to newly joined peers that have no content to share. Hence, this mechanism is introduced to help the newly joined peers.

first to increase the content availability (e.g., such a rarest-first policy is used in the BitTorrent protocol [1]).

As mentioned above, the latter architecture (mesh-based) resembles the BitTorrent protocol. BitTorrent was developed for distribution and replication of large content files with the help of peer resources. BitTorrent uses file segmentation and the content that is to be distributed is split into pieces (chunks). There are no fixed data paths for all chunks since every chunk follows a different route to arrive at peers. The transfer of chunks can be concurrent. Concurrent flows among peers can be bi-directional, used for uploading, downloading, or both. BitTorrent can be used to provide not only file sharing but also P2PTV/P2PVoD streaming and actually this protocol is the most common mechanism of commercial P2PTV/P2PVoD applications available today. This type of P2P systems (BitTorrent-like mesh-based) is our main focus in this thesis when referring to P2PTV and P2PVoD.

1.2 Quality of Experience

A valuable measure of a system and the offered service is users' satisfaction on its performance and quality. Quality of Experience (QoE), also known as "Quality of User Experience", is the term used to describe end users' perception. In the fields of Information Technology and Telecommunications, QoE is a subjective measure of a customer's experience of a multimedia service, e.g. Voice-over-IP, TV, VoD, Video Conferencing, Gaming, etc.

QoE measurements help to quantify the quality improvements, disputes, and repairs as well as monitor the quality, when designing and evaluating multimedia delivery systems.

Many factors determine the overall experience at an end user, like the video quality, the delay, the reliability and robustness of the system, etc.

QoE is related to, but differs from, Quality of Service (QoS), which attempts to objectively assess the service delivered. QoE represents the subjective measure, but can also be assessed objectively using three types of models: Full-reference Model, Partial-reference Model, and No-reference Model. The No-reference model has no knowledge of the original stream or source file and tries to predict QoE by monitoring several QoS parameters in real-time. The Partial-reference model has some limited knowledge of the original stream and tries to combine this with real-time measurements to reach a prediction on the QoE. The Full-reference model assumes full access to the reference video by comparing the original video before the transmission and the received video after the transmission. The Full-reference model gives the best accuracy, but usually is done offline and only applied in the case that the tester has control over both the sender and the receiver. The Full-reference model is also the one used in our QoE measurement study.

1.3 Streaming Systems

Streaming can be defined basically as a method of transferring digital data that carries real-time characteristics in such a way that the recipient can view the content while receiving data. A client application, known as streaming player, can start playing back streaming media as soon as enough data has been received without having to wait for the entire file to have arrived. As data is transferred, it is temporarily stored in a buffer until enough data has been accumulated to be properly assembled into the next sequence of the media stream. The main advantage of streaming is that content of any length, even live content of unlimited length (i.e., webcamera content), can be played by the end user. The main disadvantage of streaming is that the playback quality depends on the network bandwidth. Poor network conditions and bandwidth fluctuations easily result in annoying disruptions.

This thesis is partly based on the analysis of different streaming systems: 1) TV streaming systems, further divided in P2PTV, IPTV and CDN; 2) Video-on-Demand Streaming Systems; and 3) Video Conferencing Systems.

1.3.1 Part I: TV Streaming Systems

P2PTV

The P2PTV we consider in this thesis is a chunk-based video streaming system (like PPlive [2], Coolstreaming [3], and PPStream[4]). It does not use application-layer multicast trees, because they currently do not work well in a dynamic environment when peers arrive and depart frequently.

The core operation of P2PTV is that, in a limited time period, every node downloads the near-future video content, while displaying the existing content stored in a buffer. These active peers exchange the video content, depending on the periodically updated data availability information.

The differences between P2PTV and the P2P file-sharing systems (e.g. BitTorrent mentioned in Section 1.1) are basically threefold: (1) P2P file sharing has no strict time limitation so that it may be blocked only if none of the downloaders have the remaining chunks. However, P2PTV may face much more blocking when a user can not find available chunks or download them fast enough; (2) In P2P file sharing, after finishing downloading the whole file, the peer will change from a downloader (uploading at the same time of downloading) to a pure seed (no downloading, only uploading). However, P2PTV peers can only be downloaders, not pure seeds because one P2PTV peer only has two actions: one is opening the P2PTV and watching live TV programs, the other is closing the P2PTV. In the first case, he needs continuous downloading to feed the video display. At the same time, he may upload the video chunks in his buffer to

---

[4] http://www.ppstream.com/

other peers. However, the time of keeping a chunk is not infinite because the peer has to give up old video chunks to release his memory space. Hence, in the first action, the peer downloads while uploading. In the second action, as soon as he switches off the P2PTV, he will stop both downloading and uploading; (3) P2PTV systems have departure pulses due to many peers leaving immediately and simultaneously at the end of programs. But in P2P file sharing, peers leave mostly after they finish the download. They have asynchronous completions of file downloads.

We will now turn to describe how P2PTV works.

A TV program streaming content is divided into chunks, each of which consists of a fixed amount of content (e.g. 10 Kbytes). All video chunks in this TV channel are made available from an original source provider.

We introduce seven basic steps of a newly joined node registering into the P2PTV community and setting up connections with its partners for viewing a particular TV channel.

1. User gets a list of TV channels from a P2PTV website.

2. User selects a particular channel.

3. Once the channel is selected, the user registers with a bootstrap root server and gets a list of hosts currently watching the same channel.

4. User chooses a subset of active hosts on the list to establish a partner group for his downloading.

5. In the partner group, peers exchange the “Buffer Map” (BM). A BM message indicates which chunk a peer currently has buffered and can share. For instance, assume each chunk contains 1-second of video, a sliding window of 60 chunks can effectively represent the buffer map of a node. A BM can be recorded by 60 bits, with 1 indicating that a chunk is available and 0 unavailable.

6. After establishing the partnership and collecting the BMs, the user may download chunks from these partners simultaneously and also may upload the chunks in his buffer to these partners, depending on the BMs he gets. A partner can be the parent, the child, or both, of the user. If there are more than two partners having a same chunk, then the user can choose the partners using a peer selection policy[5] (e.g. choosing the partners with higher upload bandwidth, enough available time, etc.). In the partnership group, they periodically exchange the data availability information (BM) to retrieve the necessary chunks continuously. Besides updating

---

[5] For example, as indicated in [3], the computation overhead for deciding which chunk should be downloaded from which partner is quite low, only about 15 ms in their experiments.

the BMs periodically, the user can also continuously search for better partners to establish new partnership[6].

7. After downloading and buffering some chunks, e.g. 10 seconds of video, the P2PTV software will launch a media player to display it. While displaying the video content already in the buffer, it needs to download the near future video fast enough.

The P2PTV system will be investigated and analyzed in Chapters 2 and 3.

IPTV

Figure 1.1 gives an overview of the physical configuration of a common type of IPTV system (for example, KPN's Mine TV[7] in the Netherlands). Another type of IPTV systems, like Swisscom's Bluewin TV[8] and Microsoft's IPTV[9], combines IPTV service and Internet access and the bandwidth is not dedicated to a specific service.

In this thesis, the first type of IPTV system, which has a dedicated infrastructure and bandwidth, is our main focus.

In the infrastructure network in Figure 1.1, video programming, broadband Internet data, and telephone calls are delivered, where IPTV uses IP multicast and uses the service provider's specific broadband network to deliver TV programs. We can see from Figure 1.1 that, in the core network, regular telephony service is implemented in the public voice system, while IPTV video data is transmitted in a dedicated IP network specialized for IPTV service and Internet data is delivered via the Internet network (the Internet is an IP network, but the IP network does not only represent the Internet). All these three services are joined at the DSLAM (digital subscriber line access multiplexer) in the access network. Users connect to the DSLAM with about 1 to 5 km copper wire, on which DSL is applied.

Focusing on the IPTV service, the television programs are collected at a TV data centre. This TV data centre is connected to many source providers so that more and more TV programs can be collected easily from all over the world. The IPTV programs are encoded using the MPEG4 codec and sent out over the IP network (fully controlled by the service operator). The servers running IPTV software are in constant communication with the users' home set-top boxes, via the DSLAMs at local telephone central offices.

---

[6] For instance, as indicated in [2], one loop time of probing (probe all active peers to find new active peers and then form a partner group) is 6 seconds in their measurement study.

[7] http://www.kpn.com/prive/tv/interactieve-tv.htm

[8] http://www.swisscom.com/GHQ/content/Media/Medienmitteilungen/2005/20050526_03_bluewin_tv.htm?lang=en

[9] http://news.cnet.com/Swisscom-to-test-Microsofts-IPTV/2100-1026_3-5102162.html

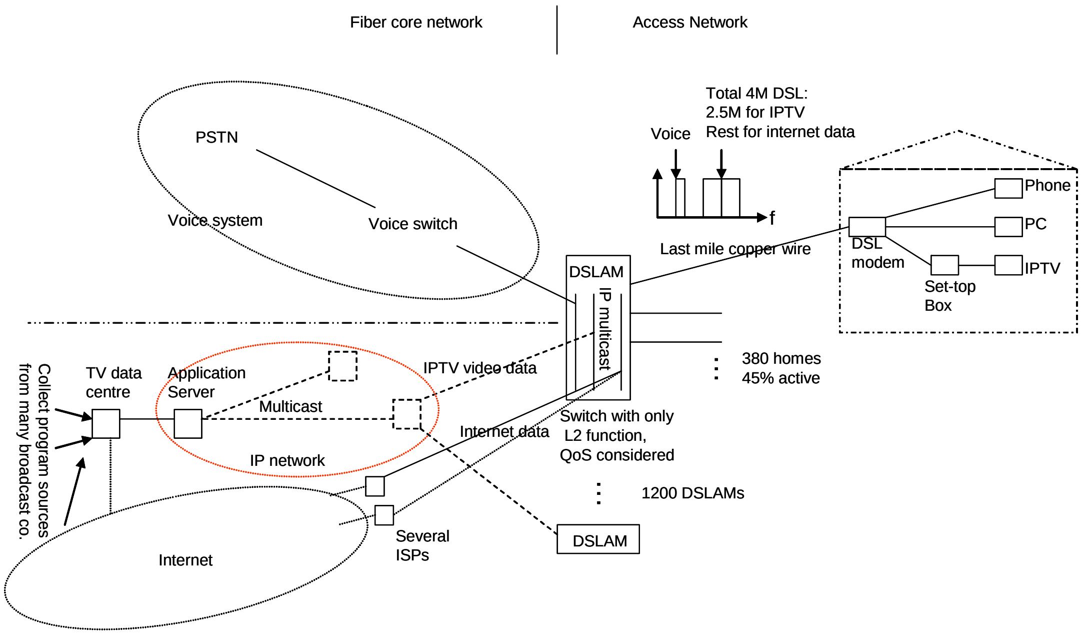


Figure 1.1: IPTV configuration

Standard IPTV contains embedded software that enables to initiate and receive television through the data network using protocols such as IGMP and SIP. IGMP version 2 is used for connecting to a multicast stream (TV channel) and for changing from one multicast stream to another (TV channel zapping). SIP can establish sessions.

An IP layer multicast tree is used to deliver the live TV programs to viewers. The DSLAM is facing the most replication workload, because it directly connects the end users with the leaf link of the multicast tree.

This multicast IPTV system will be analyzed in Chapter 4.

CDN

A content delivery network (CDN) is a system of computers containing copies of data, placed at various points (caches) in a network so as to maximize content availability for clients. Caches[10] are used to reduce bandwidth requirements, reduce server load, and improve the client response time for content.

Caching architectures can basically be divided into [4]: (1) hierarchical, (2) distributed, and (3) hybrid architectures.

In the case of a hierarchical caching architecture, the original complete content

---

[10] A cache is equipped with a hard disk of sufficient capacity and with digital recording hardware and software.

sources are stored in a central server located at the highest level of a caching hierarchy, while Residential Gateways (RGs) at home will act as caches at the lowest level of a caching hierarchy [5]. The content could be stored not only at the home on a RG, but also in the caches placed at multiple levels in the hierarchical network [6]. A hierarchical caching architecture is particularly beneficial when cooperating cache servers do not have high-speed connectivity. In this case, popular objects can be efficiently diffused toward the demand. On the other hand, hierarchical caching has several drawbacks, compared with the other two caching architectures: (1) setting up a cache hierarchy requires caches to be placed at the key access points in the network, which requires significant coordination among the participating cache servers, (2) a caching hierarchy may introduce additional delays, (3) high-level caches may become performance bottlenecks, and (4) multiple copies of the same object may be stored at different cache levels, which is inefficient.

In distributed caching systems, there are only caches at the bottom level, and there are no intermediate caching levels. To decide from which cache to fetch an object, the caches need to keep track of metadata information about the content of the caches. Consequently, with distributed caching most of the traffic flows through the low network levels, which are less congested, and no additional disk space is required from the intermediate levels.

In a hybrid caching scheme, caches may cooperate with other caches at the same level or at a higher level using distributed caching.

For web caching, Rodriguez *et al.* [34] performed a numerical analysis of hierarchical and distributed caching. In their work distributed caching achieves shorter transmission times (i.e., the time to send a document from the cache to the destination) than hierarchical caching, and has very good performance in well inter-connected areas without requiring intermediate cache levels. However, the deployment of distributed caching on a large scale encounters problems, such as large connection times (i.e., the time that a request travels to hit a document in the distributed or hybrid caching architectures), high bandwidth use and administrative issues. Therefore, the CDN system discussed in this thesis is a hierarchical network which combines the multicast IPTV service and the unicast cached Video-on-Demand streaming service. It will be analyzed in Chapter 5.

1.3.2 Part II: Video-on-Demand Streaming Systems

For the Video-on-Demand (VoD) service, the simplest way to transfer a multimedia file is bulk file transfer. The content is simply a common file of known size available at a source. Every client first downloads the complete file and then can reproduce its multimedia content.

Differently, the multimedia content could be delivered in the form of a media stream. The content is not distributed as a complete file anymore, but delivered as a continuous ordered flow of data from a source. We call it VoD *streaming* service, which is also our main focus in this part of the thesis.

Different from the live TV streaming service, in the VoD streaming system the viewers can view the media content starting from any point of the video (i.e. from the beginning of the video or from its middle). When a user wants to reproduce and display the content, it contacts the source that contains the stream. The biggest challenge for this service is the start-up delay, which is the time a user has to wait after a request is ordered to start reproducing the content. However, this service has no strict real-time requirement like live TV streaming, since users can wait some time before the content starts to play.

On-demand streaming is to deliver multimedia content of a known size that is available at a source node. There are several ways to realize the delivery. For example, the content could be stored at a server and a client downloads it in the displayed order from the server as he is viewing it. Otherwise the content could be spread in a P2P way.

1.3.3 Part III: Video Conferencing Systems

Video Conferencing (referred to as VC in the following text) conducts a conference between two or more participants at different sites by using telecommunications to transmit audio and video data. During the video conference, not only the camera video streaming content, but also the documents, computer-displayed information and whiteboards can be shared among participants.

Standards of VC

Three main umbrellas of standards for VC[11] by the International Telecommunications Union (ITU) are listed:

1. ITU H.320 is known as the standard for running Multimedia (Audio/Video/Data) over Integrated Services *Digital* Networks (ISDN) [7]. ISDN is a circuit-switched telephone network system. It specifies technical requirements for narrow-band visual telephone systems and terminal equipment, typically for videoconferencing and videophone services. Previously, video conference systems with H.320 as a basis were common. Business, government, and military organizations predominantly use H.320. Today video gateways can be used for communication between the converged H.323 network and the legacy video network.

2. ITU H.324 is an ITU-T recommendation for voice, video, and data transmission over regular *analog* phone lines. The H.324 standard is formally known as terminal for low bit-rate multimedia communication. H.324 covers the technical

---

[11] http://en.wikipedia.org/wiki/Videoconferencing

requirements for very low bit-rate multimedia telephone terminals operating over the Plain Old Telephone Service (POTS). H.324 bridging and conversion services make video conferencing easier by using existing phone lines. The media-enabled 3G mobile phone has caused the creation of a derivative of the H.324, which is called H.324M (3G-324M).

3. ITU H.323 [8] is a recommendation that defines the protocols to provide audio-visual communication sessions on any *packet* network, such as LANs and Internet. The H.323 standard addresses call signaling and control, multimedia transport and control, and bandwidth control for point-to-point and multi-point conferences. While H.323 excels at providing basic telephony functionality and interoperability, H.323's strength lies in multimedia communication functionality designed specifically for IP networks. It is widely implemented by voice and video conferencing equipment manufacturers, and is used within various Internet real-time applications (e.g. Microsoft NetMeeting).

For example, for digital lines Cisco video conference solutions [9] use H.323 and have a H.320 gateway connect to an ISDN network and also support third-party Session Initiation Protocol (SIP) [10] Proxy.

Technology of VC

According to the study of VC on the Internet, two core technologies of VC are explained in this section. More technologies like Media Synchronization using Real-time Transport Control Protocol (RTCP), traffic classification and shaping using the Diff-Serv (differentiated services) model to provide the appropriate QoS guarantees to video traffic, will not be discussed here.

1. Video Compression: In VC, audio and video data must be transmitted in real time. Therefore, a high bandwidth is required. But it is not enough to only have high bandwidth. For example, "true color" needs 24 bits per pixel. A full screen image might be 640x480 pixels, over 7 million bits. For full motion video, the image is refreshed 25 times per second. This adds to over 184 Mbit/second. It is not desirable to transmit the information at this rate on the Internet, so video compression is required. The hardware or software performing the compression is called a codec. The mostly used video codec standards in VC are H.261, H.263, and MPEG-4, where H.261 and H.263 codecs were developed by the ITU, and MPEG-4 were developed by ITU-T together with the ISO/IEC Moving Picture Experts Group (MPEG) [11]. Nowadays, compression rates of up to 1:500 can be achieved.

2. Data Delivery: The most used ways of data delivery in VC are (1) analog or digital telephone network (POTS or ISDN), (2) LAN or Internet, and (3) Satellite.

> Firstly, for the approach (1), ISDN is offered by many telephone companies that provide fast, high-capacity digital transmission of voice, data, still images and full-motion video over the worldwide telephone network. ISDN is rapidly growing in popularity and is widely accepted in industry as the way to access multimedia over a network. Although it is still expensive when compared to a standard line, particularly for primary rate access, it may be suitable for inter-site conferencing. Secondly, for the approach (2), Internet Protocol (IP) is the protocol used for communicating data across the Internet. ITU approved the H.323 transmission standard in 1996, and Session Initiation Protocol (SIP) was approved by IETF in 2002. Videoconferencing over IP has become more widely accepted with each passing year. Finally, for the approach (3), Satellite transmission is usually used for one-to-many conferences. Although it is expensive, cost will not be affected by distance. For meetings involving a larger group, or for those taking place in venues with little or no connectivity, satellite video conferencing is the ideal solution.

A survey about VC can be found in Chapter 8.

Part I

TV Streaming Systems

Chapter 2

P2PTV: Measurement Study

Increased Internet speeds together with new possibilities for tailor-made television services have spurred the interest in providing television via the Internet. Recently, there has been a growing interest in academic and commercial environments for live streaming using P2P technology. A number of new P2P digital Television (P2PTV) applications have emerged. These P2PTV applications have been developed with proprietary technologies. Their traffic characteristics and the Quality of Experience (QoE) provided by them are not well known. Therefore, investigating their mechanisms, analyzing their performance, and measuring their quality are important objectives for researchers, developers and end users.

In this Chapter, we will focus on an existing popular P2PTV application, called SopCast. We will investigate its working mechanisms and evaluate its performance via measurement studies.

2.1 Introduction

The success of peer-to-peer (P2P) BitTorrent[1] file sharing is undisputed. Their idea of exchanging fragments has also been applied to streaming applications over a peer-to-peer network. In recent years, many such peer-to-peer video streaming applications, e.g. CoolStreaming[2], PPLive[3], Tribler[4] and SopCast[5], have appeared and are receiving much attention. Measurements on these systems show that more than 100,000 concurrent users viewing a single channel is not uncommon [2]. In this section, we will investigate a P2PTV system called SopCast [12]. In order to understand the mechanisms of this

---

[1] http://www.bittorrent.com/

[2] http://www.coolstreaming.us/hp.php?lang=nl

[3] http://www.pplive.com/en/index.html

[4] http://www.tribler.org/trac/wiki

[5] http://www.sopcast.com/

BitTorrent-based P2PTV system and its performance, we will investigate by means of measurements the functionality and the traffic characteristics of SopCast and the Quality of Experience (QoE) perceived by its end users. QoE can be measured through objective and subjective measurements.

2.2 Related work

There are numerous P2PTV applications available: e.g. SopCast, CoolStreaming/DONET [3], Joost[6], PPlive [2], PPstream, TVAnts [13], Tribler [14], etc. However, only few measurement studies were performed on P2PTV.

Hei *et al.* [2] have measured PPLive via passive packet sniffing. Their measurement study focused on three important aspects of PPLive: streaming performance, workload characteristics, and overlay properties. They presented detailed session statistics, such as session duration, packet size and the correlation between them, and traffic breakdown among sessions. Start-up times and video buffer dimensions were also presented.

Ali *et al.* [15] evaluated the performance of both PPLive and SopCast. They collected packet traces of the systems under different conditions and analyzed the data on a single host joining a system and then tuning into a channel, and collected packet traces for these cases.

Silverston and Fourmaux [16] analyzed the different traffic patterns and underlying mechanisms of several P2PTV applications. During the 2006 FIFA World Cup, they collected packet traces from PPLive, PPStream, SopCast and TVAnts. This work differs from [2] by the number of applications studied and the followed comparative approach. Results in this study are based on a single day where two soccer games were scheduled. The measured download traffic indicates that the applications use different mechanisms to obtain the video and in addition they maintain a different peer neighborhood.

Most of the previous work is executed from a single point of observation [16], or from few nodes within direct access [15] and lacks an automatic mechanism for conducting measurements. Also, the final perception of the end user, i.e. the Quality of Experience, is not taken into account. In our opinion, it is important to investigate the Quality of Experience for P2PTV systems, since P2PTV technology can be considered a promising candidate for content distribution companies to deploy flexible and interactive TV. In this Chapter we perform such a study, through objective and subjective measurements, for the P2PTV application SopCast.

---

[6] http://www.joost.com/

2.3 How does SopCast work?

SopCast is a free BitTorrent-like P2PTV application, born as a student project at Fundan University in China. The bit rates of TV programs on SopCast typically range from 250 Kb/s to 400 Kb/s with a few channels as high as 800 Kb/s. The channels can be encoded in Windows Media Video (WMV), Video file for Realplayer (RMVB), Real Media (RM), Advanced Streaming Format (ASF), and MPEG Audio Stream Layer III (MP3).

The SopCast Client has multiple choices of TV channels, each of which forms its own overlay. Each channel streams either live audio-video feeds, or loop-displayed movies according to a preset schedule. The viewer tunes into a channel of his choice and SopCast starts its own operations to retrieve the stream. After some seconds a player pops up and the stream can be seen. SopCast also allows a user to broadcast his own channel.

In this Section, we will investigate the SopCast P2PTV system by answering the following question: *What is the operational mechanism in SopCast?*

The full description and more analysis on its topological properties can be found in [17].

2.3.1 Experimental settings

In order to have a controlled environment to better understand the target packets functionalities and communication patterns, we have used PlanetLab[7] as our experiments platform. The experiments consist of two types of nodes:

1) A standard personal computer located in our campus network, which acts as the *source provider*[8] (SP). With the SP, we registered a dedicated channel to the SopCast network. In this channel, a small cartoon movie with a duration of 2 minutes and size of 3.5 MBytes is continuously broadcast in a loop. Thus, our experiment resembles a streaming system. The SP runs Windows XP. It is equipped with an Intel Pentium 2.4 GHz processor, 512 MB RAM and a 10/100 FastEthernet network interface, which is further connected through a router to the Internet.

2) The second type of nodes are PlanetLab nodes that act as SopCast peers viewing the TV channel released by us. Each of the 51 PlanetLab nodes under consideration runs the following software: (1) SopCast Client (Linux version), with command line control; (2) TcpDump[9] to enable passive monitoring of the traffic transmitted

---

[7] http://www.planet-lab.org/

[8] The source provider is the node who broadcasts the entire video by using SopCast software.

[9] http://www.tcpdump.org/

at the SopCast peers; and (3) Perl[10] Scripts to remotely control the PlanetLab nodes to have some actions or collect and record the data.

Thus, our experiment resembles a streaming system, as shown in Figure 2.1.

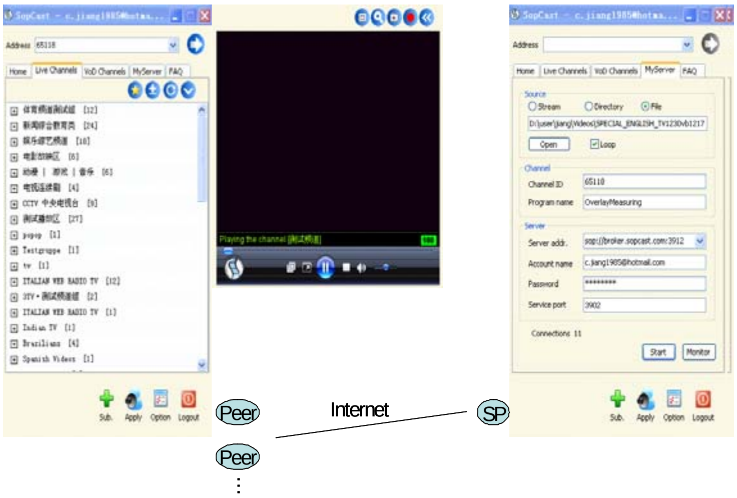

Source: ○ Stream ○ Directory ● File
D:\user\jiang\Videos\SPECIAL_ENGLISH_TV1230vb1217
☑ Loop
Channel ID: 65110
Program name: OverlayMeasuring
Server addr.: sop://broker.sopcast.com:3912
Account name: c.jiang1985@hotmail.com
Password: ********
Service port: 3902
Connections 11

Figure 2.1: The SopCast player at the Peer side (left); The window of the SP interface (right).

Passive monitoring by its nature is limited to information acquired from the communications that are visible to the monitoring stations. By accessing all of our PlanetLab nodes, we attempt to capture data that is as complete as possible and use it for our characterizations.

We conducted a single experiment on 11:00 am, August 22^{nd}, 2008. The experiment lasted for roughly 40 minutes. The 51 PlanetLab nodes were controlled in such a way that they joined and left the network simultaneously. We collected the traffic log files captured by TcpDump from all the 51 peers. Traffic collection at the SP was accomplished with Ethereal [18]. The collected trace files from the 51 PlanetLab nodes were further processed and analyzed with AWK[11] scripts.

2.3.2 Dissecting the SopCast protocols

SopCast is a proprietary P2PTV application and consequently the SopCast website only provides limited information about SopCast's video delivery mechanism. Although the

---

[10] http://www.perl.org/

[11] AWK is a general programming language that is designed for processing text-based data, either in files or data streams.

work presented in this Section has been conducted in a thorough and careful way, without having the source code of SopCast, the claims that we have made are based on our investigation of SopCast. They are not the exact description of the protocol. In this Section, we only present our conclusions on the peer communication scheme and video delivery rule in Sopcast.

Identification of SopCast packets

TcpDump reveals that SopCast relies on UDP. Since we are not able to decode the SopCast traffic, it is not possible to tell exactly what kind of messages are being exchanged in the captured trace files. To figure out the packet functionalities, we studied the packet lengths and the corresponding delivery patterns. Table 1 presents our findings.

Table 1. Summary of SopCast traffic

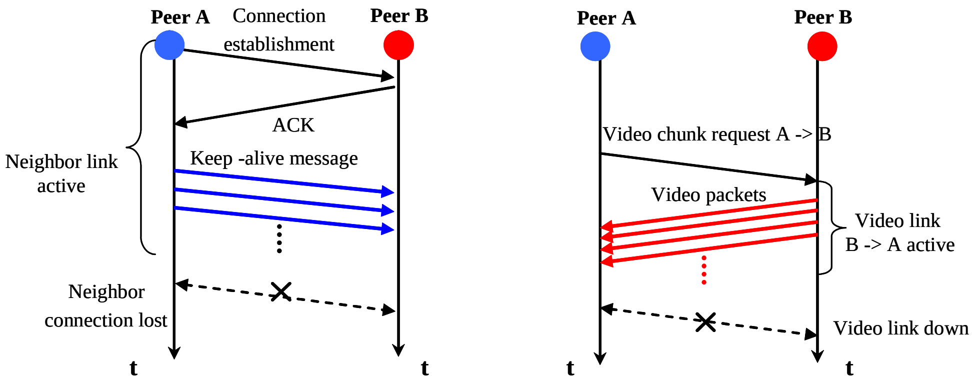
Type	Size (bytes)	Functionality
Video	1320	Maximum size of the video packets
packet	377, 497, 617, 1081, 1201	Video fragments
	52	HELLO packet to initiate link connections
	80	Confirmation on receiving the HELLO packet
Control	28	Acknowledgement
packet	42	Keep-alive message with neighbors
	46	Video requesting packet

We have captured the traffic at peers and also analyzed how two peers communicate with each other and set up the video transmission session, which will be described in the following two sections.

Neighbor communication in SopCast

Assuming that a SopCast peer has retrieved a random list of peers (a *peerlist*) in the network, it will start to choose some peers with whom connections are established. If two peers exchange 42-byte control packets with each other, we refer to these peers as being *neighbors*.

Communication between two peers in SopCast is always initiated by a $52 - 80$ byte packets pair. Once the connection is established, the pair of peers keeps exchanging a sequence of 42-byte packets with each other. The 42-byte packets are transmitted with a high frequency, roughly every second. We denote this packet as a *keep-alive* packet. With the keep-alive packets, a peer announces its existence in the network. The purpose is to accommodate overlay dynamics, and maintain neighbor relation with others via the decentralized communication between peers. Peers can lose neighbors. In case a neighbor does not respond to the keep-alive packets, the peer stops contacting



(a) Peer communication (b) Video delivery between peers

Figure 2.2: Diagram of (a) neighbor communication and (b) video delivery in SopCast.

this neighbor until it chooses the neighbor again. A graphic illustration of the neighbor communication scheme is shown in Fig. 2.2(a).

Video delivery in SopCast

Video delivery in SopCast is chunk-based. The TV content in SopCast is divided into *video chunks* or *blocks* with equal sizes of 10 kbyte. This finding is in line with the video chopping algorithm implemented in many existing P2P systems, although the chunk size may be different[12]. If a peer is providing video packets to its neighbors, we refer to the peer as a *parent*. A *child* is defined if a peer is receiving video packets. A peer is allowed to have multiple parents and multiple children. A parent-child relation can be established only when two peers are neighbors. A peer is free to request multiple video blocks from its parents.

A peer in SopCast never voluntarily delivers video streams to its neighbors. To download video packets, a child always needs to request them from its parent(s) via a *video request packet* with the size of 46 bytes, see Fig. 2.2(b). After receiving the request, its parent(s) will deliver a series of video packets to the child. In case the child needs more blocks, it sends another request.

In the trace, we noticed that the non-video packets with 46 data bytes are transmitted periodically. Within the transmission of two consecutive 46-byte packets, a sequence of video packets with 1320 data bytes are sent to and acknowledged (using a non-video packet with 28 data bytes) by another peer. After the 1320-byte packets sequence, there is a smaller-sized video packet (with $377, 497, 617, 1081$ or 1201 data bytes) following at the end for making up a rounding size of one or multiple video chunks (we observed

---

[12] In Bittorrent, the default size of the chopped block is 256 kbytes.

that a video chunk size is equal to 10 Kbytes). The SopCast traffic pattern during a video session between any two peers is shown in Figure 2.3.

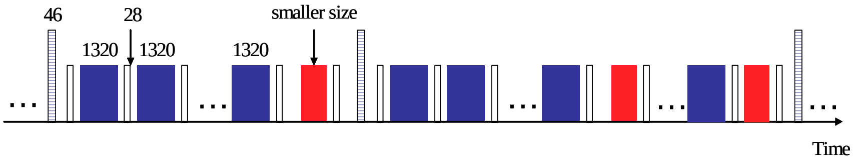


Figure 2.3: Traffic pattern during a video session between 2 peers.

The reason to have this traffic pattern is due to the IP fragmentation principle. The requested blocks are treated as a large datagram during transmission, and this large datagram (such as 10 kbytes) should be segmented into smaller pieces in order to pass over the Ethernet. SopCast sets the maximum size of the video packet to be transmitted to 1320 bytes. A generalization of the video block fragmentation rule is

$$n \times 10 \ kbyte = x \times 1320 \ bytes + y \tag{2.1}$$

where n is the number of requested video blocks, x is the number of 1320-byte video packets, and y is the size of the smaller fragment in bytes, as presented in Table 1.

Transport protocol

The reports of Wireshark revealed that SopCast relies on UDP traffic. We have observed two peaks in the packet size distribution: one falls in the region below 100 bytes and another one at 1320 data bytes. The small packets with less than 100 bytes are considered non-video packets, which are used for application-layer acknowledgments of data packets delivered, requests for video chunks, or initial connection establishment. The bigger packets, with size approximately equal to the Maximum Transmission Unit (MTU) for IP packets over Ethernet networks, are the video packets.

We also observed that SopCast faces a high overhead, about 60% of non-video packets versus almost 40% of actual video data packets (see Figure 2.4). This was expected since the protocol works on top of UDP, which does not guarantee reliability in the way that TCP does. For time-sensitive applications, UDP is a reasonable choice, because dropped packets are considered no worse than delayed packets. However a minimum control on the status of the chunks must be kept. Since the chunks arrive out of order, a scheme is needed to keep track of the video chunks that need to be reassembled in order and buffered, and in case a chunk is missing, to retrieve it. Nevertheless, various small packets are exchanged among peers to keep the peer list up-to-date, to test the status of peers (e.g., is enough bandwidth available) or to distribute the chunk availability information and the keep-alive messages. This explains the overhead in this kind of mesh-based P2PTV system.

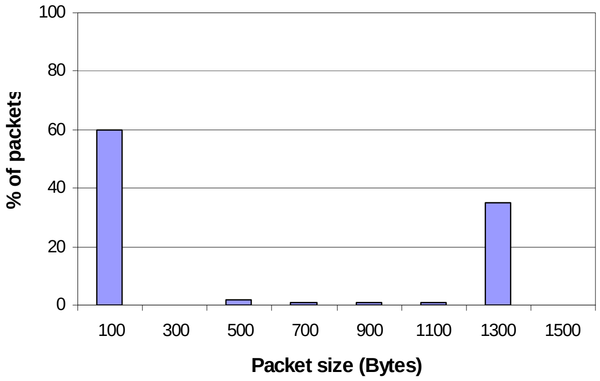


Figure 2.4: SopCast packet size distribution.

Peer exchange and architecture

When SopCast first starts, it requires some time to search for peers and subsequently it tries to download data from the active peers. We recorded two types of start-up delay: the delay from when one channel is selected until the streaming player pops up, and the delay from when the player pops up until the playback actually starts. The player pop-up delay is in general 20 to 30 seconds. This is the time for SopCast to retrieve the peer list and the first video packets. The player buffering delay is around 10 to 15 seconds, which can vary from player to player and is not related to SopCast. Therefore, the time that passes for a user to enjoy the live streaming ranges between 30 and 45 seconds.

Examining the traffic generated by each node we found that the first task of each viewer node is sending out a query message to the SopCast channel server to obtain an updated channel list. This server has been identified, with an IP locator, to be located in China. After a peer selects one TV channel to watch, it sends out multiple query messages to some root servers (trackers) to retrieve an online peer list for this TV channel.

After contacting the tracker, the nodes form a randomly connected mesh that is used to deliver the content among individual peers. The content of a TV channel is divided into video chunks, each with equal size. A video chunk is delivered from a parent to a child peer. Except for the source, each peer in the overlay has multiple parents and multiple children. The delivery is performed with pull requesting by child peers, meaning that the chunks that a node has are notified periodically to the neighbors. Then each node explicitly requests the segments of interest from its neighbors according to their notification.

Buffering techniques

Received chunks are stored in the SopCast buffer. The buffer is responsible for downloading video chunks from the network and streaming the downloaded video to a local media player. The streaming process in SopCast traverses two buffers: the SopCast buffer and the media player buffer, as shown in Figure 2.5.

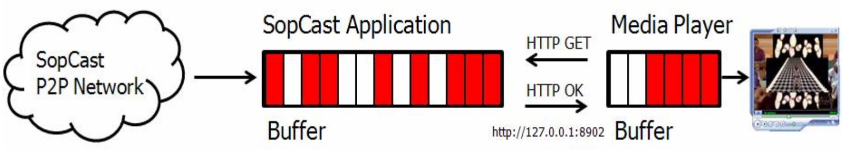


Figure 2.5: The SopCast buffer.

When the streaming file length in the SopCast buffer exceeds a predefined threshold, SopCast launches a media player, which downloads video content from the local Web server listening on port 8902. Most media players have built-in video buffering mechanisms. After the buffer of the media player fills up to the required level, the actual video playback starts.

2.4 Quality of Experience

After discussing about how SopCast works in Section 2.3, we will now investigate how it performs. In this Section, we present Quality of Experience related results [12] from a measurement study of SopCast P2PTV clients, using both objective and subjective measurement technologies. The results obtained in our study reveal the characteristics and important design issues of SopCast, as well as the QoE that the end users perceive. The measurement scenario is a global one.

2.4.1 PlanetLab Experiments Results

Measurement set-up

We have used a standard personal computer located in our campus network, as the source provider (SP) of a TV channel content. With the SP, we registered and broadcasted a dedicated TV channel to the SopCast network. In this channel, a video with 480*384 resolution and at 45 KB/s is continuously broadcasted in a loop.

On the other hand, we have used scripts not only to remotely control 70 PlanetLab nodes (as our Peers) to view the TV channel we released, but also to monitor the QoE at them. Therefore, a bit different from the experiment settings in Section 2.3.1, each of the 70 PlanetLab nodes under consideration runs not only SopCast, TcpDump, Perl Scripts, but also[13] *VLC* to capture the stream.

We make use of traced files of this SopCast network captured during 10 months (May. 2007 - Nov. 2007; Aug. 2008 - Oct. 2008). In particular, we collected the traffic logs for several one-hour intervals from the 70 peers under investigation.

Upload and Download rate

Comparing the video data upload and download rates (the rate here is the average value over one hour trace), we noticed that only few nodes have higher upload rate compared to their download rate. In Figure 2.6 the four nodes that have higher upload than download rates have been identified as “supernodes”.

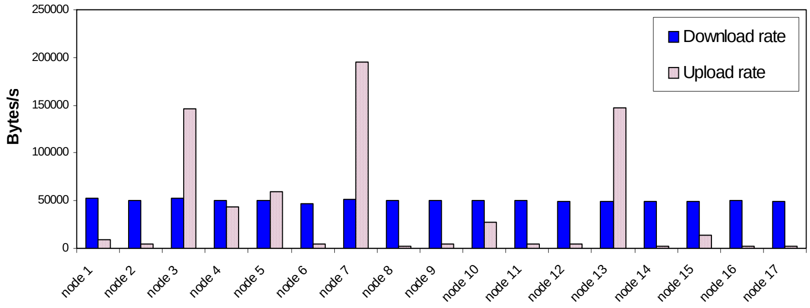


Figure 2.6: Upload and download rates of the peers at 17 of the 70 PlanetLab nodes.

The average download rate at each node is almost the same. This suggests that the download rate at a peer seems to be confined by SopCast.

Parent’s upload rate to one child

As mentioned in Section 2.3.2, we define a parent as a peer that is uploading video packets and define a child as a receiver of video packets.

The best choice for a peer is to download from the parent who has enough “parent upload rate” per peer (the rate is the average value over one hour trace). However, from Figure 2.7 it can be seen that the majority of the parents keeps the same amount of upload rate per peer, which is approximately 24 KB/s. This behavior does not change with the addition of more peers.

Based on the results of Figures 2.6 and 2.7, we can imagine that a parent with larger upload rate probably has more children than a parent with smaller upload rate.

---

[13] VideoLan-Client: http://www.videolan.org

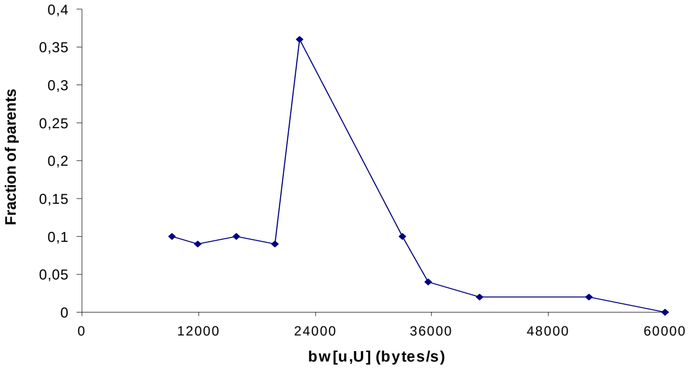


Figure 2.7: Parent's upload rate per peer when the network size is 70. u represents a parent and U represents a child of the parent.

Blocking

We first monitored the download rate without any buffering. We compared the fluctuating download rate with the steady playback rate of the video (45 KB/s). If the download rate is smaller than the playback rate, the end user faces blocking or video freezing because no data is buffered. However, due to the fact that the buffering technology is widely used in practice, the result of this experiment can be considered as a worst-case study [19].

The worst-case blocking probability is calculated counting the time t_{block}, where the download rate is smaller than the playback rate, divided by the total amount t of the observed time.

$$Pr[block] = \frac{t_{block}}{t}$$

Based on our calculation, we found that, without buffering, blocking happens during 22% of the time. Such a value is too high for a smooth playback, which illustrates the significance of a buffer. It also indicates that the existing mesh-based P2P technology cannot provide a truly real-time streanming service with acceptable blocking probability.

The scenario without a buffer represents that the data will be processed as soon as it arrives at the destination. However, in a P2P network there is no real data path with continuous data flow transmitted in order. Different chunks are delivered via different paths and the arrival time of the chunk is also based on the chunk availability at that moment. According to the data driven mechanism of SopCast, a buffer is needed to map video chunks and based on which peers can request what they need.

Sentinelli *et al.* [20] observed that the SopCast buffer contains one minute of video. We made the assumption that the media player uses a buffer of m seconds, where m is usually smaller than 10. When an end user starts up a SopCast TV channel, basically once the SopCast buffer is filled up, it injects m seconds of streaming content into the media player buffer, as depicted in Figure 2.5. By the time the media player consumes those m seconds of video, SopCast is downloading new video packets to refill the buffer.

If the SopCast buffer fails to collect enough data to feed the media player buffer, blocking occurs.

In Figure 2.8 the buffer behavior of one peer is depicted.

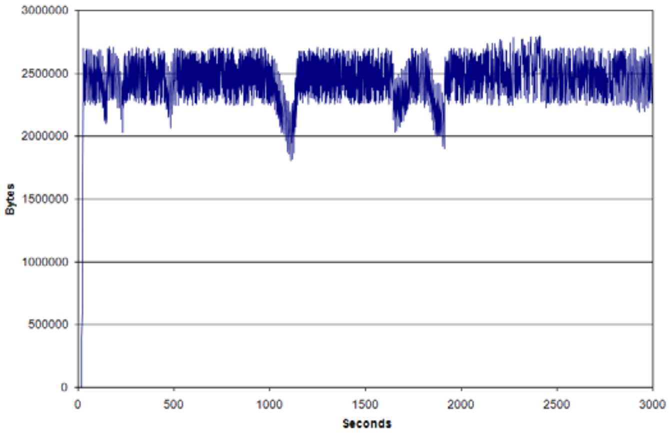


Figure 2.8: SopCast buffer content in bytes of node planetlab1.diku.dk.

We consider the SopCast buffer size as the streaming rate of the video (45 KB/s) times one minute [20], equal to about 2700 Kbytes, which can be seen in Figure 2.8. We can observe that after the start-up phase, the buffer maintains stable and the playback is continuous. The average download rate for this node is with 127 KB/s far higher than the streaming rate of the video (45 KB/s). Hence, it was expected that blocking would not happen. However, due to fluctuation of the download rate with time, the data stored in the buffer has major drops in the intervals between 1040 - 1150 s, 1660 - 1730 s, 1840 - 1920 s. During these drops (meaning that in these periods the data stored in the buffer is much less than the full buffer size of 2700 Kbytes), end users *may* face blocking (e.g., image freezing or loss), because in the worst case, the lacked video chunks may be the ones which need to be displayed in the next m seconds.

Overall video packet loss

As mentioned in Section 2.3, every peer can be downloading data from other peers and at the same time be uploading data to others. For instance, we have 4 peers viewing our TV channel, PlanetLab nodes A, B, C and D. After analyzing the trace file of node A, we know that he downloads data from nodes B, C and D during the whole trace. We can calculate how much video packets A received from B by analyzing the trace file of A, as well as how much video packets B sends to A by analyzing the trace file of B. Then in the video session between A and B, we can get its packet loss ratio (same for the video sessions between A and C, and between A and D). To summarize the number of packets lost in all video sessions of receiver A, we can get the overall video packet loss ratio at node A during the whole trace. The same approach is applied to the other nodes and the distribution of the overall video packet loss ratio is plotted in Figure 2.9.

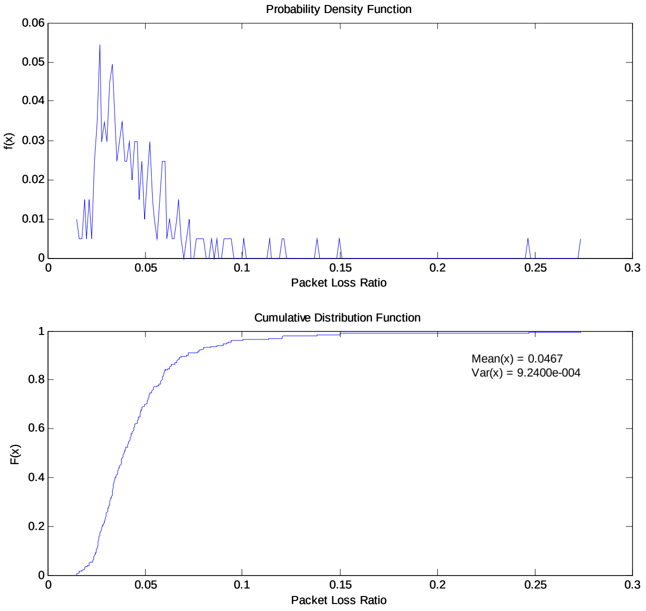


Figure 2.9: The overall video packet loss ratio during the whole trace.

In Figure 2.9, the x axis represents the overall packet loss ratio during the whole trace at an end user and the y axis represents the percentage of end users in our network. The mean value of the packet loss ratio at an end user is over 4%, which is high compared to the baseline SDTV packet loss ratio requirement in IPTV of 0.4% [19]. Besides, we observed that the packet loss at a peer is mainly caused in the beginning period of this peer entering this TV channel network (maybe because the connections between him and his parents are not optimized or stable yet). However, thanks to the buffer, this high video packet loss does not have much affect on the video quality, which can be seen in Section 2.4.1.

Video Quality

Here, we assess the video quality at the end user with respect to their start-up freezing time, overall frame loss ratio, image quality and audio-video synchronization.

Start-up freezing time Many video decoders use “copy previous” error concealment to hide missing frames in the video stream from users. This means that in the event of not receiving a certain frame, the last correctly rendered frame is displayed on the screen, resulting in the frame freezes that are often seen in Internet video playback.

Through our experiments on PlanetLab, we observed that 97.17% of nodes always first face a freezing image for a period of time at the beginning of viewing the TV channel. The reason could be that the node has just started downloading chunks from other peer nodes and the video buffer of the local SopCast webserver is created but not filled enough for the media player to access, or the lost or delayed packets cause undecodable frames. Therefore in the beginning of viewing the TV channel, there usually occur many frame losses with frame loss ratio approaching 100%. During this period, the media player handles the position of dropped frames by displaying the nearest good frame (the first good frame in this case) as a stagnating picture. The duration of this period (the start-up freezing time) indicates, after the user sees the first image, for how long (s)he has to wait before the video playback starts playing smoothly.

We used[14] *VirtualDub* to investigate the beginning of the captured WMV files at each node. Figure 2.10 shows that very few end users face a start-up freezing time of more than 10 seconds. The peak appears at 1 second (about 18% of the nodes experience 1-second freezing time when they start viewing the video). On average, end users see the first freezing image for 4.13 seconds before seeing the actual stream of the TV channel.

This result indicates the importance of improving the current scheduling and decoding technologies of SopCast.

Overall frame loss The frame loss discussed here only considers lost frames, not damaged frames (frames downloaded partially with some packets lost) after the start-up freezing phase.

In Figure 2.11, the x axis represents the overall frame loss ratio during the whole trace at an end user and the y axis represents the fraction of end users in our network.

The mean value of the frame loss ratio is 0.82%. It means that end users could have a good video quality with low frame loss ratio after the start-up freezing time.

---

[14] *VirtualDub* is a video capture and video processing utility for Microsoft Windows.

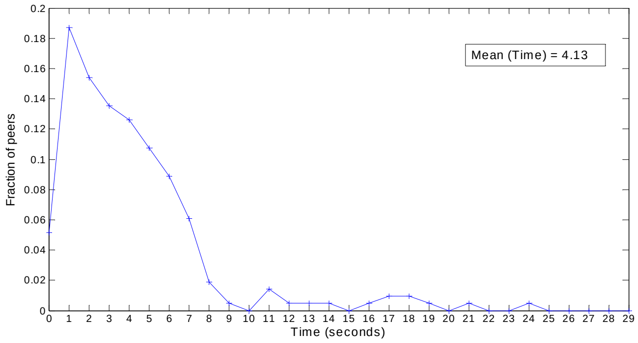


Figure 2.10: Start-up freezing time.

Image Quality After the start-up freezing phase of peers, we cut the received video at them and synchronized each frame of the cut received video with the cut original video to get the average objective Mean Opinion Score (MOS) [21], using bVQM [22].

bVQM (Batch Video Quality Metric) is a software tool developed by the Institute for Telecommunication Science to objectively measure perceived video quality. It measures the perceptual effects of video impairments including blurring, jerky/unnatural motion, global noise, block distortion and color distortion, and combines them into a single metric.

bVQM takes the original video and the processed video and produces quality scores that reflect the predicted fidelity of the impaired video with reference to its undistorted counterpart. To do that, the sampled video needs to be calibrated via *VirtualDub*. The calibration consists of estimating and correcting the spatial and temporal shift of the processed video sequence with respect to the original video sequence. The final score is computed using a linear combination of parameters that describe perceptual changes in video quality by comparing features extracted from the processed video with those extracted from the original video. The final score is scaled to an objective MOS value, a measure for user perceived quality, defined on a five-point scale; 5 = *excellent*, 4 = *good*, 3 = *fair*, 2 = *poor*, 1 = *bad*. MOS here does not take audio quality, zapping time, etc. into account.

We captured at selected nodes the stream retrieved from the SopCast buffer with VLC.

We broadcasted two videos at different data rates: one at 45 KB/s (the most common data rate used in SopCast) and another one at 1 Mb/s. The scores provided by bVQM are plot in Figure 2.12.

The minimum threshold for acceptable quality corresponds to the line MOS = 3.5. The average MOS are high for both streaming rates, only a negligible degradation has been observed. This also suggests that SopCast does not provide any kind of encoding

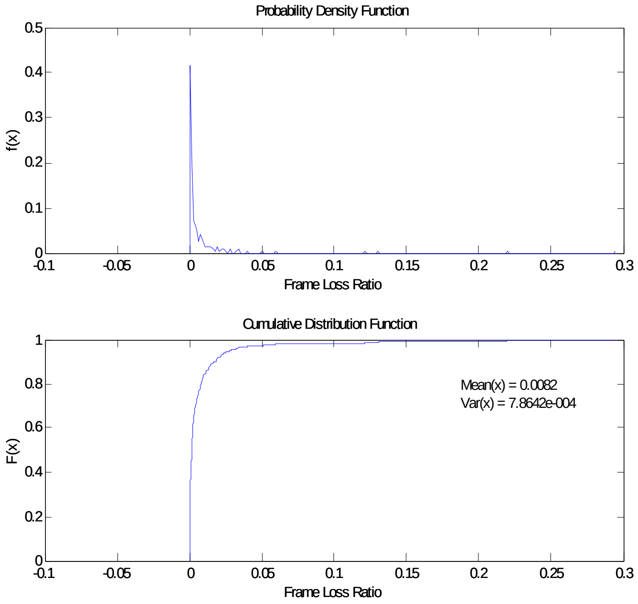


Figure 2.11: The overall frame loss ratio during the whole trace.

to the broadcasted video.

Audio-Video Synchronization

Audio-video synchronization refers to the relative timing of sound and image portions of a television program, or movie.

The International Telecommunications Union [23] recommendation states that the tolerance from the point of capture to the viewer/listener shall be no more than 90 ms audio leading video to 185 ms audio lagging behind video.

We decided to analyze the A/V synchronization in SopCast with an "artificially generated" video test sample. The test sample includes a video component and an audio component. The video component and the audio component both contain a marker. The video marker displays between a first video state and a second video state, a red full screen image. Similarly, the audio waveform alternates between a first audio state and a second audio state, an audio "beep". The video and audio waveforms are temporally synchronized to transition from one state to another at the same time.

The video is broadcasted with SopCast. When the audio and video tracks were extracted and compared, it turned out that there was an average difference in time between the two tracks of about 210 ms, which exceeds the ITU recommendation. The reasons are twofold: (1) We believe that the main contribution to this time shift is caused by the network. When the video is sent into the network, due to its transport

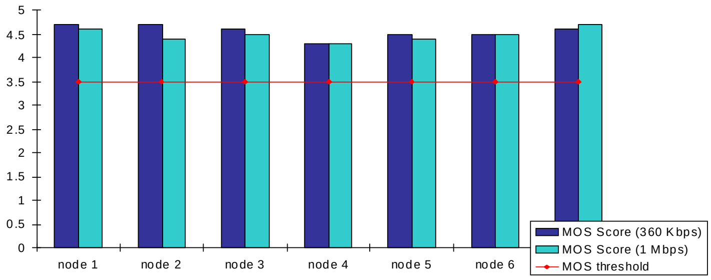


Figure 2.12: Objective MOS for the received videos.

protocol (UDP), some packets might get lost. Since the system is displaying in real time, a loss of a video packet can cause the decoder to adjust buffer allocations affecting the synchronization of audio and video tracks. (2) The direct digital-to-digital conversion from one (usually lossy) codec to another. We needed to convert from the original video format to the streamed one, passing through a final reconversion of the received file to extract the tracks. This (re)conversion may also have affected the synchronization.

Peer Synchronization

While watching a football match it could be disturbing to hear the neighbors scream "GOAL" while still watching the pre-goal action [20]. Such phenomena are common in P2PTV systems and are referred to as peer lags. While watching the same channel, peers' content might not be synchronized. We measured the different lag delays by injecting in the SopCast network another artificial video that mainly reproduced a timer. Each second a sequential number is shown. Since SopCast builds a webserver that feeds the player's buffer, we connected 6 instantiations of VLC to the webservers of the representative nodes and we gathered the visualization on a PC, see Figure 2.13.

Clearly, some peer's content lags behind that of others. In the environment of PlanetLab, the lag went up to 3 seconds. In reality, the lag is expected to grow even further. Hence, we can conclude that SopCast clients are not likely to view an exactly same frame of the stream at the same time. We can say that SopCast nowadays is not yet suitable to distribute football-like content due to the low synchronization level among users.

Zapping Time

While watching TV a common behavior is to change from one channel to the other, the so-called "zapping". If P2PTV applications want to gain popularity in the field of home entertainment it is necessary to look at the zapping performance of P2PTV applications. While for analog TV, zapping consists of scanning through different television channels

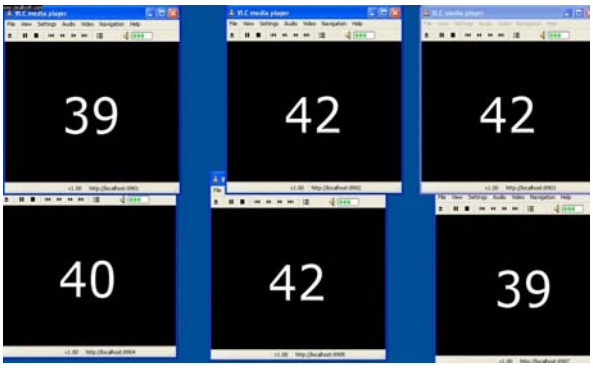


Figure 2.13: The video at different nodes.

or radio frequencies, in P2PTV the initial list of hosts must be retrieved, and the system tries to connect to some of the hosts to get data.

To measure the SopCast zapping time we needed to calculate the time that SopCast requires to fill its buffer and build the local web server. To do that we developed a Perl script that starts a counter when a channel is clicked and it stops when enough data to be displayed has been fetched.

We let the script run when zapping among 20 popular and less popular channels. Figure 2.14 shows the distribution of the zapping times. It turns out that the zapping time in SopCast is very high.

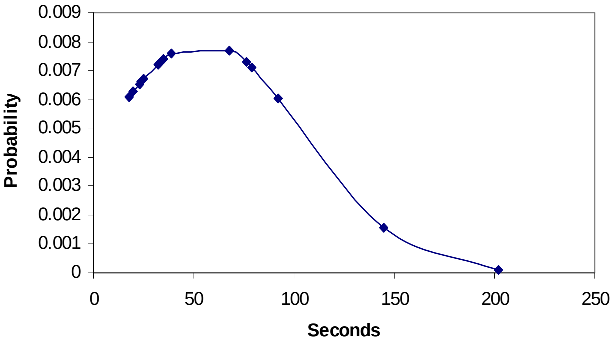


Figure 2.14: Distribution of zapping time.

Changing channels in an analog TV network usually takes about $\frac{1}{2}$ to 1 second compared to Digital TV where zapping times of more than 2 seconds might be experienced. Note that according to the DSL Forum the zapping time should be limited to a maximum of 2 seconds [24]. In the IPTV environment changing channels or zapping, has great importance as this is very often regarded as the most important parameter used to judge the overall quality of the network seen from the end user perspective. With an average zapping time of 50 seconds, SopCast (P2PTV) faces an unacceptable delay. Customers expect information being delivered to their screen as soon as possible. Hence, much improvement is needed in the start up phase of SopCast.

2.4.2 Subjective Measurements

Subjective video quality is concerned with how video is perceived by a viewer and designates his or her opinion on a particular video sequence. Subjective video quality tests are quite expensive in terms of time and human resources. To evaluate the subjective video quality, a video sequence is chosen. Under typical settings of the system, the sequence is presented to the users and their opinions are collected. The opinions are scored and an average value is computed.

Approach

The following steps were used for the subjective evaluation:

- 22 persons participated in the evaluation by viewing SopCast TV channels and completing a questionnaire.
- The questionnaire contained 10 questions each addressing the expected quality problems of SopCast.

The 10 questions were:
(1) How fast was the login process?
(2) How long did you have to wait before seeing the stream after you started the channel?
(3) How long did you have to wait before seeing a stable stream?
(4) Was the size of the video screen satisfactory (resolution, stream bit rate)?
(5) During the observation period, did the video unexpectedly stop?
(6) Did you observe any bad frames in the video (a bad frame refers to a mosaic-like image)?
(7) Did you observe any freezing frames in the video (a freezing frame refers to a brief stop, say a second, in the video playback after which it resumes to a normal playback)?
(8) How was the voice quality (cuts, clarity, volume) of the channel?
(9) Were the audio and video synchronized throughout the playback time?
(10) Are TV channels provided by SopCast interesting and are the amount of TV channels enough?

The questionnaire used the standard MOS scale. The subjective MOS does not only consider the quality of video, but also the start-up time, the extent of the usage convenience, and the feeling about the TV channel content itself.

- Every question had a weight (the weights of the questions are also decided by the participants) depending on the severity of the issue and its influence on the QoE of SopCast. Based on the weight given to each question, the overall MOS per questionnaire was calculated as follows:

  $$MOS = \frac{\sum_{x=1}^{10} Weight_x Score_x}{\sum_{x=1}^{10} Weight_x}$$

  where $Weight_x$ represents the weight of question x and $Score_x$ represents the score of question x.

Result

The MOS over all the participants is 4.08 (see Figure 2.15). This means that the channel's video quality is good. The subjective MOS score is and was expected to be lower than the objective score in Section 2.4.1, because more measures than only video quality play a role.

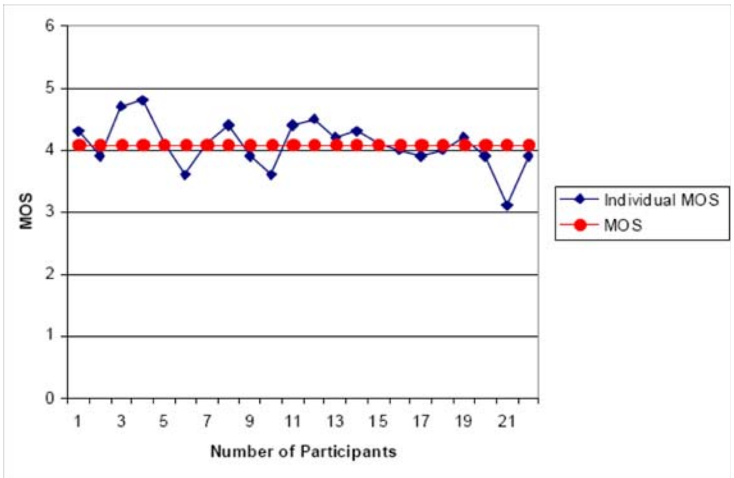


Figure 2.15: Subjective MOS scores.

2.4.3 Conclusions

The aim of this work was to understand, with a series of experiments, the behavior of a popular P2P streaming system called SopCast. Through passive measurements, we characterized SopCast's behavior and evaluated users' QoE.

Based on our measurement results on the traffic characteristics of SopCast, the main conclusions are: (1) There is a lot of overhead in the form of non-video packets; (2) The average video download rate is almost the same at each peer; (3) Peers' upload rates differ substantially, but the majority of the parents keeps the same amount of upload rate per peer; (4) In the worst case, a peer will face video blocking very frequently, but the situation can be much improved with the help of buffers; (5) Overall packet loss ratio is high.

For QoE metrics, in other related works, researchers usually only look at the video quality when making claims on the QoE. However, in our work we have shown that more measures should be taken into account, such as the blocking, the audio-video synchronization, synchronization level among peers, the TV channel zapping time, etc. Based on our measurement results on the QoE of SopCast, the main conclusions are: (1) SopCast can provide good quality video to peers: low overall frame loss ratio and high MOS scores; (2) Audio and video for SopCast can be out-of-sync, and may even exceed the requirements from the ITU; (3) SopCast suffers from peer lags, i.e. peers watching the same channel might not be synchronized; (4) The zapping time in SopCast is extremely high.

The innovative measurement methods and scripts mentioned in this Chapter can also be applied to other measurement studies and for other streaming applications.

Chapter 3

P2PTV: Analytical model

Disregarding economic incentives, the TV content delivery service operator should consider to deploy P2PTV technology if connecting end-users' set-top boxes in a P2P way can achieve better Quality of Experience (QoE) than the tranditional IPTV (Server-to-Client) infrastructure. Hence, defining, computing and comparing the quality of these two television architectures is deemed essential.

In this Chapter, we will specifically investigate one important QoE measure, namely the content blocking probability, via a mathematical model. We will still focus on the mostly used BitTorrent-like P2PTV system, and analyze its performance in terms of blocking probability. Our P2PTV case study is based on measurements of SopCast, discussed in the previous Chapter.

3.1 Related Work

Related work on BitTorrent-like P2P blocking and performance [25], [26], [27], only targets the P2P file sharing systems. Empirical blocking results were provided [2], [3], but a definition of and a formula to compute the end-to-end blocking in P2PTV architectures are still missing. In this Chapter, we provide a first step in this direction.

3.2 Model with assumptions

Figure 3.1 presents our model of chunk-based P2PTV applications. We focus on the blocking of a single television channel i for a particular user U. User U is viewing a television channel i. The playback rate of the channel i is v kbits/s. In the worst case, user U downloads the next second of content when displaying the current second of content. One second of television content is divided into R chunks. Hence, one chunk contains v/R kbits.

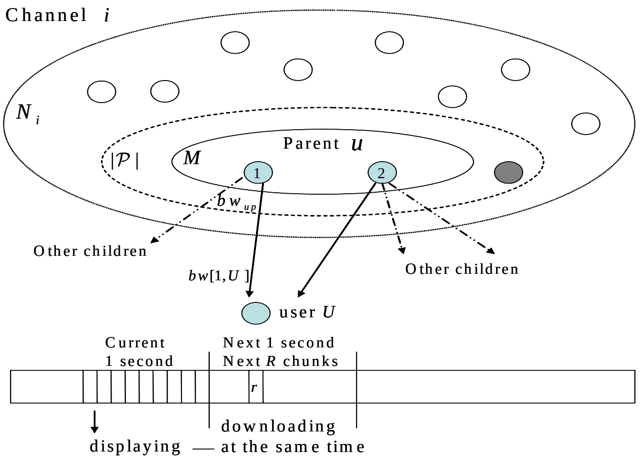


Figure 3.1: P2PTV model when user U has 2 parents.

First, based on a peer list that user U obtains, a fixed amount of peers (referred to as partners) are randomly chosen to form a partner group $\mathcal{P}$ for that channel i. In this partner group, partners exchange information about which chunks are available at which partner. Based on this chunk-availability information, a user U chooses $M \le |\mathcal{P}|$ peers from whom he downloads chunks. Those peers are called parents of user U. A parent supplies at least 1 chunk to user U. Similarly, the peers downloading the content from user U will be referred to as children of user U. A parent has Y children at the same time. We denote by N_i the number of available peers in the entire peer list corresponding to channel i and by bw_{up} the upload bandwidth of a parent. Simultaneously while downloading the chunks, user U is displaying the content stored in his buffer[1]. Here, we consider the *worst case* of P2PTV blocking where none of these R chunks were previously downloaded and the download of these R chunks has to be finished within the coming second.

3.3 Computation of the blocking probability

A P2PTV network built on top of the Internet has, in theory, more available capacity contributed by peers than the traditional Server-Client network. Hence, the P2PTV blocking is mainly caused by content unavailability, peer selection randomness and peer dynamics. Blocking in P2PTV can also occur while watching television. We define the blocking probability in P2PTV as the probability that user U cannot successfully download the next R chunks before having displayed the current R chunks.

---

[1] There is a zero probability of buffer overflow, because the buffer window is sliding and the outdated chunks will be deleted automatically.

If the required R chunks cannot all be found in the partner group, which happens with probability $b_{chunk}(i)$, blocking will occur. Even when all the chunks can be found at the $|\mathcal{P}|$ partners, a chosen parent may upload his chunks too slowly. This occurs with probability $b_{time}(i)$. Finally, if a parent leaves during uploading, blocking will occur. The probability of this occurrence is $b_{dyn}(i)$. As a reasonable approximation, we assume that the different events are independent. The end-to-end blocking $b(i)$ of channel i in P2PTV can be presented as:

$$b(i) = b_{chunk}(i) + (1 - b_{chunk}(i))(b_{time}(i) + (1 - b_{time}(i))b_{dyn}(i)) \tag{3.1}$$

In the sequel, we determine $b_{chunk}(i)$, $b_{time}(i)$ and $b_{dyn}(i)$.

$b_{chunk}(i)$: Choosing parents among partners is based on the chunk-availability information. Blocking arises when user U cannot find all R chunks from the *randomly chosen* partners. We let $1 - b_{chunk}(i)$ denote the probability that user U can find all R chunks in his partner group.

$$1 - b_{chunk}(i) = \prod_{r=1}^{R}(1 - B_i(r)) \tag{3.2}$$

where $B_i(r) = [1 - \pi_i(r)]^{|\mathcal{P}|}$ represents the probability that user U cannot find chunk r successfully among $|\mathcal{P}|$ randomly chosen (and hence considered independent) partners. We use $\pi_i(r)$ to represent the probability that a peer is storing chunk r.

$b_{time}(i)$: Blocking arises when at least one parent u cannot upload his chunks to user U within 1 second due to insufficient bandwidth. The upload bandwidth that a parent u has available for user U is $bw[u, U]$, which has a distribution function $\Pr[bw[u, U] \leq x]$. Given that user U requests X_u chunks of v/R kbits from parent u, the required upload bandwidth is $bw_{X_u} = \frac{X_u v}{R}$. If the available bandwidth is smaller than the bandwidth required for uploading the requested chunks, blocking will occur.

In the following, we will make use of the theory of partitions [28]. A partition of a positive integer R is a collection of positive integers whose sum is R. If there are M terms in the sum, then R is said to be partitioned into M parts. We let $p_M(R)$ represent the number of partitions of R into parts not exceeding M, which has the generating function [28]: $\prod_{j=1}^{M}(1 - x^j)^{-1} = \sum_{R=0}^{\infty} p_M(R)x^R$. From the generating function the following recursion is obtained:

$$p_M(R) = p_{M-1}(R) + p_M(R - M)$$

with $p_M(0) = p_1(R) = 1$ and $p_M(j) = 0$ for $j < 0$.

In our P2PTV model, we need to find the number of partitions of R chunks over exactly M parents, which equals $p_M(R) - p_{M-1}(R) = p_M(R - M)$. The occurrence probability of having M parents has a density function $\Pr[M = k]$, however for partitions with a same number of M we assume that the occurrence probability of each possible partition is the same.

Blocking occurs only when the available bandwidth $bw[u, U]$ is smaller than the required bandwidth. Assuming independence,

$$b_{time}(i) = \sum_{k=1}^{|\mathcal{P}|} \frac{\Pr[M = k]}{p_k(R-k)} \sum_{j=1}^{p_k(R-k)} \left(1 - \prod_{u=1}^{k} (1 - \Pr[bw[u, U] < \frac{X_u(j)v}{R}])\right) \tag{3.3}$$

where the index j refers to one of the $p_M(R-M)$ possible partitions, and $X_u(j)$ refers to how many chunks user U requests from parent u. For instance, we have $p_M(R-M) = 2$ possible partitions of $R = 4$ chunks over $M = 2$ parents (u_1 and u_2), namely partition $j = 1$ consisting of $X_{u_1}(1) = 1$ and $X_{u_2}(1) = 3$ and partition $j = 2$ consisting of $X_{u_1}(2) = 2$ and $X_{u_2}(2) = 2$.

$b_{dyn}(i)$: We let $b_{dyn}(i)$ represent the probability that at least one parent leaves during his uploading period, which causes blocking.

In our P2PTV model, the peer departure process $Z(t)$ is considered Poissonian. According to the measurement study of X. Hei *et al.* [2, Fig 13.], the CDF of TV viewing time follows an exponential distribution, which justifies our assumption that the TV user departure process is Poissonian. The rate at which peers leave from channel i is denoted by θ_i. The departure rate of one peer is $\frac{\theta_i}{N_i}$, where N_i stands for the total number of available peers in channel i.

We denote by $P_u(j)$ the probability that parent u with $X_u(j)$ chunks (in partition j) leaves during uploading. $P_u(j) = \Pr[Z(t+t_0) - Z(t_0) = 1] = (\frac{\theta_i}{N_i}t)\exp(-\frac{\theta_i}{N_i}t)$, where $t = \min\{1, \frac{X_u(j)v/R}{bw[u,U]}\}$.

Hence, we can express $b_{dyn}(i)$ as:

$$b_{dyn}(i) = \sum_{k=1}^{|\mathcal{P}|} \frac{\Pr[M = k]}{p_k(R-k)} \sum_{j=1}^{p_k(R-k)} \left(1 - \prod_{u=1}^{k} (1 - P_u(j))\right) \tag{3.4}$$

Substituting formulae (3.2), (3.3) and (3.4) into formula (3.1), we obtain the end-to-end blocking $b(i)$ of P2PTV.

3.4 Case Study

In this P2PTV case study, we analyze the chunk-based P2PTV application SopCast. We developed scripts and set-up a distributed measurement testbed via Planetlab, as described in Section 2.4.1. We installed SopCast and TcpDump on each (of in total 80) Planetlab nodes and performed various measurements, among which we believe three are unique, namely: (1) measurements on the topology - we have the parent distribution $\Pr[M = k]$ for popular as well as unpopular channels, (2) measurements on the bandwidth distribution $\Pr[bw[u, U] \leq k]$ (see Figure 2.7), and (3) the quality of experience (e.g., blocking) at the end user. We have used our measurement data as input to our model and then computed the blocking probability according to the formulae above. We also compared our analytical results with our measurement results on blocking probability.

Our experiments with SopCast indicated that the upload bandwidth of a parent bw_{up} is almost uniformly shared by his Y children in a 1 second period of time. Hence, for user U, we can define $bw[u, U] = \frac{bw_{up}}{Y}$. Since each parent can have a different upload rate and a different number of children at a given time, the value of $bw[u, U]$ may differ per parent. Later, we will use our empirically obtained $bw[u, U]$ distribution function into our model.

We assume that there are K available television channels that can be viewed. The channel popularity distribution α_i for the channels $(1 \leq i \leq 23)$ is obtained from a market survey[2]. We assume that the remaining $K - 23$ channels uniformly share a popularity of 5.9%. In reality, the channel popularity distribution changes more smoothly, but here we just classify channels as popular or unpopular.

We let $N_i = N\alpha_i$, where N is the number of concurrent active peers over all channels. N_i can be $205200\alpha_i$ (the same size for IPTV for a fair comparison later) or $2,200,000\alpha_i$ (the current peak size of a P2PTV system). For other values, our case study is based on SopCast. Our measurements were run for 5 times on 5 different days and we have used the obtained results for our computations. Based on our measurement data, a minimum of 1 and a maximum of 3 partners are chosen as parents for both popular channels and unpopular channels in 1 second. The distributions of the number of parents $\Pr[M = k]$ differ for popular and unpopular channels, as shown in Figure 3.2.

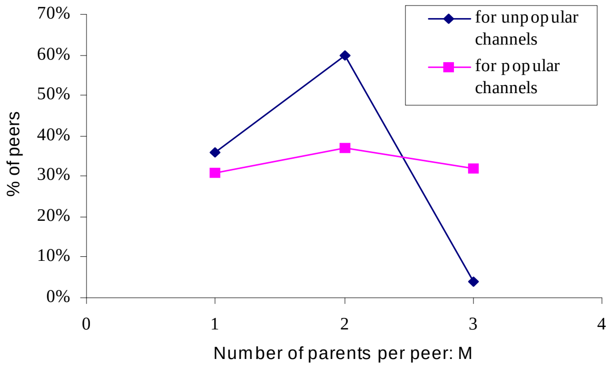


Figure 3.2: The distribution of the number of parents per peer, for popular and unpopular channels.

Our measurement study has also revealed that the channel under study has a playback rate $v = 300$ kbits/s and one chunk size is 10 kbytes (see Section 2.3.2). Hence,

---

[2] Channel 1 has a popularity of 15.1% and channel 23 of 0.2%. The 23 channels cover in total 94.1%.

R is around 3 chunks/second. In our computation of the blocking probability $b_{chunk}(i)$, we assume $\pi_i(r)$ equals to 91% ([29]) for each channel. We set $v/R = 104$ kbits, to retrieve the blocking probability $b_{time}(i)$. We assume that the download bandwidth of an end user is large enough to download $R = 3$ chunks in 1 second. In order to obtain the blocking probability $b_{dyn}(i)$, we define $Q_{P2PTV} = \frac{\lambda}{\theta_{i.}}$ and further assume[3] $\theta_i = 0.00038N/Q_{P2PTV}$, with θ_i fixed for each TV channel. λ represents the arrival rate of users into the system.

Given the above values we can get $b_{chunk}(i)$, $b_{time}(i)$ and $b_{dyn}(i)$, and subsequently compute the P2PTV blocking $b(i)$ from formula (3.1). Figure 3.3 plots the P2PTV blocking probability $b(i)$ as a function of the channel index i. Figure 3.3 illustrates that the 23 most popular channels have a much smaller probability to be blocked than the remaining unpopular channels. The blocking probability of all channels is quite high, because what we have modeled is the worst case without any positive effects due to buffering. In order to validate our model, we measured the blocking at end users, which was already discussed in Section 2.4.1. We monitored the download speed at each user using 1 second as unit and we compared the download speed with the playback rate. The fraction of time the download speed is smaller than the constant playback rate gives the blocking probability defined in our model. When averaging the blocking probability over all channels, we find a blocking probability of 22% when excluding the

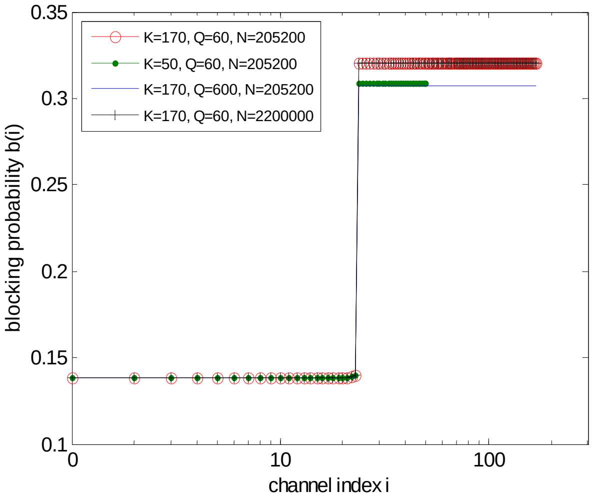


Figure 3.3: P2PTV end-to-end blocking $b(i)$ with different values for the number of channels K, the number of active users in the system N, and $Q_{P2PTV} = \lambda/\theta_i$.

---

[3] As observed in [3, Fig.14], when there are on average $N = 1750$ active peers, the maximum arrival rate of peers is $\lambda = 0.67$ peers/s and $\frac{\lambda}{N} = 0.00038$.

buffer effect (see Section 2.4.1), which fits our mathematical results.

The P2PTV end-to-end blocking $b(i)$ shown in Figure 3.3 is mainly contributed by $b_{time}(i)$. Hence, the limited upload bandwidth of an end user's parent distributed to him mainly causes the large blocking. This clearly indicates the importance of the parent selection policy.

We can also observe that a user will face less blocking if the number of available channels K is smaller, or if users leave infrequently (high Q_{P2PTV}), while changing the amount of users did not significantly affect the blocking probability.

Chapter 4

IPTV: Analytical model

After having computed the content blocking probability of P2PTV in Chapter 3, we will compute the blocking probability of IPTV in this Chapter.

4.1 Introduction

IPTV is implemented in a dedicated network, which connects the end-users' television set-top boxes through Digital Subscriber Line Access Multiplexers (DSLAMs). The television programs are collected at a data centre and distributed towards the DSLAMs along an IP-layer multicast tree. A DSLAM replicates the received signal and sends it to the end users (see Section 1.3.1).

Disregarding economic incentives, telecom operators will continue to deploy and extend IPTV in the dedicated network if its quality surpasses the quality of P2PTV. In this Chapter, we focus on the IPTV blocking probability as our QoE measure of interest, and ask the following questions:

- What factors contribute to the blocking of IPTV, and how?
- Which technology, IPTV or P2PTV, incurs the lowest end-to-end blocking?
- In what situation can P2PTV offer users better QoE than IPTV?

In order to answer these questions we have developed blocking models for IPTV which can be compared with the model of P2PTV discussed in Chapter 3. We subsequently apply it to a case study as well. Our case study of IPTV is based on measurement data of an existing Dutch IPTV network.

4.2 Related work

The work on IPTV mainly focuses on designing protocols (e.g., [30]) and implementations by broadband network operators, router manufacturers, and television providers. For the performance of IPTV, video quality and packet loss were analyzed, but the end-to-end blocking of requests has not been computed from a users' point of view. Karvo *et al.* [31] set up a queuing model to calculate the end-to-end multicast blocking, but their work is not aimed at IPTV and is not based on realistic data. In this Chapter, we will define and compute the IPTV blocking probability based on realistic data.

4.3 Model with assumptions

Figure 4.1 illustrates a possible realization of an IPTV architecture. As this figure indicates, end users are connected to DSLAMs which, on their turn, are connected to an edge router. We only model the blocking of a single television channel over a single DSLAM. We assume that C represents the capacity of a link from a particular DSLAM to the edge router. The maximum number of channels that can be transmitted simultaneously over a link with capacity C is $m = \left\lfloor \frac{C}{C_o} \right\rfloor$, where C_o is the capacity of one television channel. We further assume that there are K available television channels that can be viewed. If a channel is being viewed, we say it is "on". Not all the channels are equally popular. In our model we assume that the popularity distribution α_i is given[1].

The arrival and departure processes of TV users are assumed to be Poissonian. In reality, the request arrival rate $\lambda(t)$ is non-stationary. If a popular TV program starts, the arrival rate will be high, while once the advertisements come, the average arrival rate decreases. However, we consider the average case over a specific time period. Based on a TV-users market survey[2], the arrival process is approximately Poissonian in the period of 16:00 to 22:00. According to the measurement study of X. Hei *et al.* [2, Fig 13.], the CDF of TV viewing time follows an exponential distribution, which justifies our assumption that the TV user departure process is Poissonian.

4.4 Computation of the blocking probability

In IPTV, the limited equipment processing capability and the limited available bandwidth in an infrastructure network is the main cause of IPTV blocking.

---

[1] We have used the TV channel popularity distribution in The Netherlands, which is available at http://www.kijkonderzoek.nl/ (in Dutch).

[2] Market Report of TV customers in The Netherlands, http://www.kijkonderzoek.nl/.

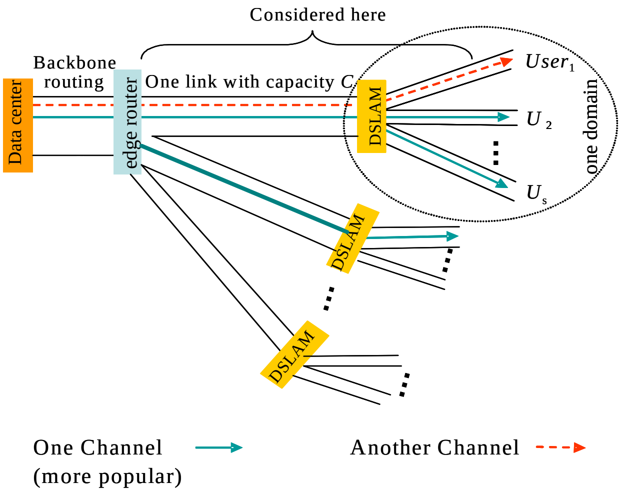


Figure 4.1: A possible realization of an IPTV architecture.

Contrary to P2PTV, where blocking can also occur while watching television, the main blocking in IPTV only occurs when requesting a television channel. If a user requesting a particular television channel i cannot receive its content, we consider channel i to be blocked. Two main causes of blocking in IPTV are:

I *Limited processing capability of a DSLAM.* If many users simultaneously desire content from the same DSLAM, blocking might occur. The blocking probability of this case is denoted as B_{proc}.

II *Insufficient available capacity from the DSLAM to the edge router.* If a maximum number of m channels is transmitted to the DSLAM, a new user requesting a new channel i (not among the transmitted m channels) will find his request blocked. Since the more popular channels have a higher probability to be present among the already transmitted m channels, $B_{link}(i)$ depends on which television channel i is requested.

Since blocking II can only occur if blocking I did not take place, for IPTV the end-to-end blocking probability $B(i)$ for channel i is

$$B(i) = B_{proc} + (1 - B_{proc})B_{link}(i) \tag{4.1}$$

B_{proc}: Assuming Poisson arrivals and departures, as explained in Section 4.3, we model the DSLAM as an $M/M/n/n/s$ queue, where s represents the number of users accessing the DSLAM and n the number of replications that the DSLAM can handle. In our model $\rho_{DSLAM} = \frac{\lambda_{DSLAM}}{\mu_{DSLAM}}$ is the same for each user, where λ_{DSLAM} represents the rate at which a user requests a TV service, and μ_{DSLAM} the rate at which the user turns his TV off. According to [32, pp. 512], we have

$$B_{proc} = \frac{\frac{(s-1)!}{(s-1-n)!n!}\rho_{DSLAM}^{n}}{\sum_{h=0}^{n}\frac{(s-1)!}{(s-1-h)!h!}\rho_{DSLAM}^{h}} \tag{4.2}$$

$B_{link}(i)$: To compute $B_{link}(i)$ we introduce two new probability functions $P(i)$ and $B_{Engset}(i)$. $P(i)$ is the probability that channel i is "on" and $B_{Engset}(i)$ is the probability that the link from the DSLAM to the edge router is consumed by m channels other than the requested channel i. Our computation of $B_{Engset}(i)$ is given in Appendix A.1, and

$$B_{link}(i) = (1 - P(i))B_{Engset}(i) \tag{4.3}$$

$P(i)$: Disregarding possible blocking[3], channel i can be modeled as an $M/M/\infty$ queue (in steady state), with infinite positions available in the queue to store all requests for channel i. Hence, the probability that channel i is "on", in the condition that no requests are blocked, is equal to the probability that the $M/M/\infty$ queue [32, pp. 281] is not empty: $\Pr[N_s > 0]_i = 1 - \exp(-\rho_i)$, where $\rho_i = \lambda_i/u_i = \alpha_i\lambda/u_i$. λ_i is the users' arrival rate in channel i, and u_i is the number of users leaving from channel i per second. λ is the users' arrival rate in the IPTV system, which includes both the rate at which users switch on their television as well as the channel switching rate. α_i represents the popularity of channel i , where $0 \leq \alpha_i \leq 1$.

Channel i can be either "off" or "on". We therefore resort to a two-state Markov chain illustrated in Figure 4.2 to analyze its steady state.

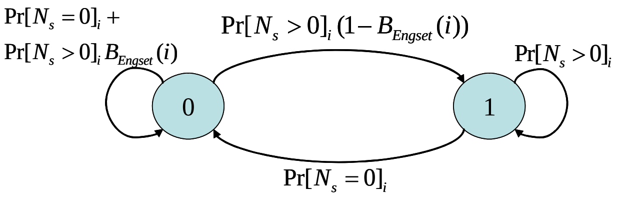

$\Pr[N_s = 0]_i +$
$\Pr[N_s > 0]_i B_{Engset}(i)$
$\Pr[N_s > 0]_i (1 - B_{Engset}(i))$
$\Pr[N_s > 0]_i$
0
1
$\Pr[N_s = 0]_i$

Figure 4.2: Two-state Markov chain representing the status of channel i, where "0" represents the state that channel k is "off" and "1" represents the state that channel i is "on".

The probability to be in the state "off" is $1 - P(i)$ and consequently the probability to be in the state "on" is $P(i)$. To change from the "off" state to the "on" state, requests

---

[3] If a user's request enters when channel i is "on", the request will be grouped into multicast. However, if a user's request enters when channel i is "off", it has to open a new channel, which might not be possible in case of insufficient available capacity.

should exist for channel i (this event has probability $\Pr[N_s > 0]_i$) and these requests may not be blocked (this event has probability equal to $1 - B_{Engset}(i)$). The process remains in the "off" state if there are no requests in the queue for channel i ($\Pr[N_s = 0]_i$) or if the requests are blocked (this event has probability $\Pr[N_s > 0]_i B_{Engset}(i)$). To change from the "on" state to the "off" state, all requests for channel i are served until the queue becomes empty (with probability $\Pr[N_s = 0]_i$). To remain in the "on" state, the queue may not be empty ($\Pr[N_s > 0]_i$).

According to the steady state probability of a two-state Markov chain [32, pp.176], the probability to be in the state "on" is

$$\begin{aligned} P(i) &= \frac{\Pr[N_s > 0]_i(1 - B_{Engset}(i))}{\Pr[N_s > 0]_i(1 - B_{Engset}(i)) + \Pr[N_s = 0]_i} \\ &= 1 - \frac{1}{\exp(\rho_i) - B_{Engset}(i)\exp(\rho_i) + B_{Engset}(i)} \end{aligned} \tag{4.4}$$

4.5 Case Study

We focus on a Dutch IPTV network for which we have obtained the following parameter values:

To allow for a fair comparison with the P2PTV case (in Section 3.4), we will choose for our IPTV case study similar values, where possible. The channel popularity distribution α_i is the same as for P2PTV, and the number of users and TV channels are the same as for P2PTV.

The link capacity C is typically 155 Mb/s in nowadays DSL systems and one MPEG4 coded TV channel consumes 2.5 Mb/s. Hence, approximately $m = 60$ different TV channels can be transmitted simultaneously to the DSLAM. We further assume that the IPTV system consists of $500,000$ subscribers, who are uniformly distributed over 1200 DSLAMs and 41% of the subscribers are active in rush hours. This results in an average number of $s = 171$ active users connecting to a single DSLAM. However, only $n = 120$ channel replications can be handled by the DSLAM simultaneously. We define $Q_{IPTV} = \lambda/u_i$.

Figure 4.3 plots the IPTV blocking probability $B(i)$ as a function of the channel index i, with $m = 60$ and $n = 120$.

The less popular channels (higher channel index) have a considerably higher probability to be blocked than the more popular channels. In addition, we can observe that the user's request is less likely blocked if there are less available TV channels (smaller K), if there are less users per DSLAM (smaller s), or if users leave more frequently (smaller Q_{IPTV}).

Figure 4.3 also plots the capacity C that is required to assure that the IPTV blocking $B(i)$, as a function of channel index i, does not exceed a certain level.

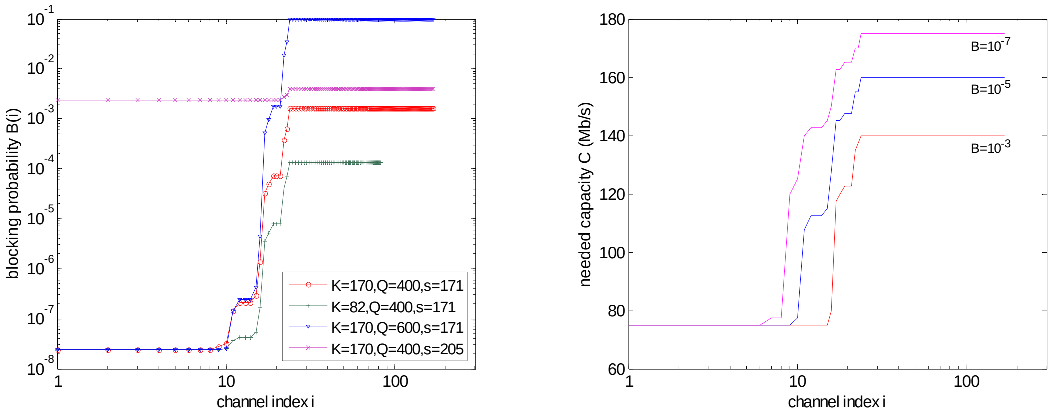


Figure 4.3: (Left) IPTV end-to-end blocking $B(i)$ with $m = 60$, $n = 120$. (right) The required capacity C for meeting different blocking requirements, with $K = 170$, $s = 171$, and $Q_{IPTV} = 400$.

4.6 Blocking Comparison between IPTV and P2PTV

We compare IPTV and P2PTV based on the computations presented in Chapters 3 and 4. We assess which technology (IPTV or P2PTV) incurs the lowest end-to-end blocking. Currently, if the operator decides to transmit television content to its residential users using P2PTV instead of IPTV, all channels will likely face more blocking than before. However, we can predict that when the number of end users increases, this will change. In a P2PTV system, the number of users has little effect on the blocking, while in an IPTV system the blocking would largely increase if the processing capability of the equipment (like DSLAM) cannot scale accordingly. When the total amount of users increases and the amount of DSLAMs and their processing capability remains the same, there will be a point at which the P2PTV system will start to outperform the IPTV system for popular channels, unless the IPTV network is extended accordingly. In our Dutch case study [19], this cross-over point is around $297,600$ users ($s = 248$ per DSLAM). Provided the appropriate data is available, our formulas allow for similar computations for different cases.

Chapter 5

CDN: Analytical model

The centrally controlled CDN system discussed here integrates the real-time TV streaming service and the Video-on-Demand service. From the analysis of Content Delivery Networking (CDN) techniques, we know that a video streaming service architecture incorporating storage at the end-user as well as in the network is a viable option for the near future.

In this Chapter, we develop a performance model for the availability of the required service. The probability that bandwidth for a required service is available is calculated as a function of video popularity, the number of available videos, cache sizes, and various parameters that characterize the network configuration and content viewing behavior. The model is applied to a MBMS (Multimedia Broadcast and Multicast Services) [35] MobileTV service with pausing function. The system performance and corresponding facility requests (e.g. cache size), and the key factors that affect the performance in different scenarios are analyzed [33].

5.1 Introduction

A content delivery network (CDN) is a system of computers containing copies of data, placed at various points (caches) in a network so as to maximize content availability for clients. Caches are used to reduce bandwidth requirements, reduce server load, and improve the client response times for content.

Caching architectures can basically be divided into: (1) hierarchical, (2) distributed, and (3) hybrid architectures, which were already explained in Chapter 1.

Based on the related work discussed in Chapter 1, we therefore concentrate our work on the use of a *caching hierarchy* model to distribute streaming content and will explain a realistic architecture in the following section.

The UMTS MobileTV network can be considered as a hierarchical architecture, hence we can apply our model to a MobileTV service. The MobileTV service investi-

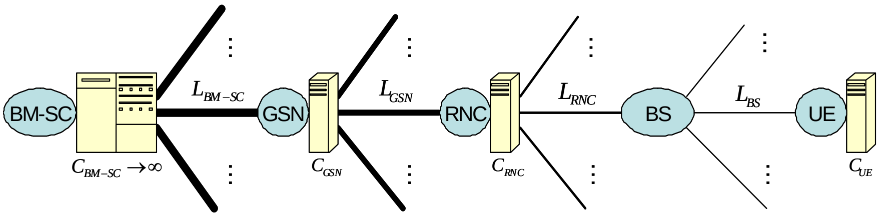


Figure 5.1: Generic model for the caching architecture to deliver video streams.

gated is a real-time live sports/news stream delivery service. It uses multicast to deliver the TV streams to save resources, which is what MBMS does nowadays. The analytical model for the multicast TV system without a pausing service can refer to [19]. Here, we consider that users can pause the TV stream, for instance when a call is coming in. Mobile Handover is considered as new user arrival/departure in our model. In reality, the feature considered for MBMS handover can be found in [36], which therefore will be not discussed here.

5.2 The service architecture

Fig. 5.1 presents a generic model for a hierarchical caching architecture. It is based on a realistic UMTS network architecture for which we assume that caches can only be placed at existing accessible locations, i.e. at locations where network equipment is already present. In Fig. 5.1, $BM-SC$ represents the highest level of a caching hierarchy which acts as a MobileTV content data center; $GGSN/SGSN$ (referred to collectively as GPRS Support Node, GSN) represents the 2nd level with Regional Exchanges; RNC (Radio Network Controller) represents the 3rd level with Local Exchanges; the BS nodes are the *Base Stations* which will not be equipped with a cache; and UE acts as User Equipment (i.e. Mobile Phone) at the lowest level of a caching hierarchy. C_i is the amount of cache disk space (in bits) at level i reserved for caching video streams, where $i \in \{UE, RNC, GSN, BM-SC\}$; and L_i represents the capacity (Mbit/s) at level-i of the link available for the video streaming service, where $i \in \{BS, RNC, GSN, BM-SC\}$. For instance, C_{UE} represents the local cache space at an end-user, L_{BS} represents the access data rate of an end-user which should not be smaller than the video streaming rate v.

To eliminate unnecessary content multiplication in the network as much as possible, the use of multicast techniques such as IP Multicast is used for content distribution when a large number of service subscribers view the same content at the same time.

We only focus on the service of delivering video streams here. The cache size and the bandwidth discussed in the following are dedicated to this service. The model is also applied to a realistic use case in Section 5.5.

5.3 Model with assumptions

We only consider streaming video applications that start from the content data center and *multicast* to all households connected to the multicast tree. The end-user has the option of zapping between videos and pausing *once* during a video. Therefore, each node must be able to store the video currently being watched in a cache connected to that node. The caches mentioned here can be located in the network. The maximum time that can be recorded depends on the cache size and the bandwidth required to deliver the video at a certain level of Quality of Service. A user may of course pause more often if a recording function on the UE allows him to do so and the UE contains enough storage capacity. However, this does not add extra blocking to the CDN as we model it (because retrieving data from the local cache will not face blocking in our model). If the user pauses more than once but the left storage capacity at UE is not enough for recording the delayed stream, he will automatically leave from the service in our model, otherwise it will cost too much network resources.

After pausing, the user can resume the same video, or zap to a different video. In the first case, the video is resumed in *unicast* mode from the closest cache which has stored and *can* provide the delayed video. In the second case, the end-user zaps to a different video, which is received via multicast.

No bandwidth is reserved for multicast and unicast separately, so multicast flows and unicast flows share the same bandwidth capacity. Hence, if too many users pause and use unicast, there will probably be insufficient bandwidth left for new multicast requests. Similarly, if users all have very different tastes and too many video/MobileTV channels are transmitted via multicast, then a new multicast request or a unicast request after pausing may face blocking due to a lack of capacity.

Here, we assume that the arrival processes of MobileTV users are Poissonian and the user viewing time is exponentionally distributed, based on regular TV user behaviors [2].

For the user behavior model, we use the following definitions:

- *Switch time*: duration that the user is watching a MobileTV channel before zapping to a different MobileTV channel. We assume it to be exponentially distributed with mean $1/\mu_s$.
- *View time before pausing*: the period an end-user watches the video stream without pausing. After this time has expired, the end-user pauses. We assume that this time is exponentially distributed with mean $1/\mu_v$.

- *View time before leaving*: the period an end-user continuously watches the video stream before leaving (excluding the pausing period). After this time has expired, the end-user leaves from the system. We assume that this time is also exponentially distributed with mean $1/\mu_L$.
- *Pause time*: the duration of the pause. We assume it to be exponentially distributed with mean $1/\mu_p$. Immediately after the pause, the end-user can have 3 options: 1) switch to another multicast video, with probability P_s; 2) turn off the MobileTV and leave from the system, with probability P_L; 3) or resume the same video stream, with probability $1 - P_s - P_L$.

Moreover, we use λ_{TV} to represent how many users, attached to one BS, turn on their MobileTV service every second; and N_{TV} represents the average number of registered MobileTV users, attached to one BS.

A registered user of our video streaming service is always in one of the following states:

- "off" state: the user is off-line, not using the service at this moment.
- "M" state: the user has started watching the video and the video is delivered via multicast.
- "$Pause$" state: the user is pausing.
- "U_{BM-SC}","U_{GSN}","U_{RNC}","U_{UE}" (unicast) states: the user continues to watch the video after having paused, and the stream is unicast from cache $BM - SC$, GSN, RNC or UE, respectively.

After pausing, whether the stream will be unicast from UE, RNC, GSN or $BM - SC$ depends on the cache sizes dedicated to the video at each level and how long the end-user paused the stream.

We define P_i with $i \in \{BM - SC, GSN, RNC, UE\}$, as the probability that the video stream is unicast from a cache at level i after a pause. $T_{UE} = \frac{C_{UE}}{v}$ is defined as the time capacity of a local cache UE reserved for a video stream, measured in the number of seconds of video that may be recorded. C_{UE} is the local cache capacity at an end user UE, measured in bits (the cache at the leaf of the hierarchy only need to store one video stream). For other levels, we set $T_i = \frac{C_i}{(\frac{L_i}{v})v} = \frac{C_i}{L_i}$, where $i \in \{BM - SC, GSN, RNC\}$, C_i is the amount of cache disk space (in bits) at level i reserved for caching several concurrent video streams (the cache at higher levels of hierarchy has to be shared by several multicast video streams), v is the video streaming rate (Mbit/s), and $\frac{L_i}{v}$ is the maximum number of video streams that can be multicast in parallel at level i. We assume that the cache at the $BM - SC$ node always contains a copy of the complete video, i.e. $T_{BM-SC} = \infty$, and $T_{BM-SC} > T_{GSN} > T_{RNC} > T_{UE}$.

The probability that the pause time is shorter than T_{UE} seconds is equal to $1-\exp(-\mu_p T_{UE})$; and the probability that the pause time is $> T_{UE}$ seconds but $\leq T_{RNC}$ seconds is equal to $\exp(-\mu_p T_{UE})-\exp(-\mu_p T_{RNC})$. Hence, P_i can be expressed in terms of the probability distribution of the duration of a pausing period as follows.

$$\begin{aligned}
&P_{UE}{=}(1\text{-}P_s\text{-}P_L)(1\text{-}\exp(\text{-}\mu_p T_{UE}))\\
&P_{RNC}{=}(1\text{-}P_s\text{-}P_L)(\exp(\text{-}\mu_p T_{UE})\text{-}\exp(\text{-}\mu_p T_{RNC}))\\
&P_{GSN}{=}(1\text{-}P_s\text{-}P_L)(\exp(\text{-}\mu_p T_{RNC})\text{-}\exp(\text{-}\mu_p T_{GSN}))\\
&P_{BM-SC}{=}(1\text{-}P_s\text{-}P_L)(\exp(\text{-}\mu_p T_{GSN})\text{-}\exp(\text{-}\mu_p T_{BM-SC}))
\end{aligned}$$

When $T_{BM-SC} \rightarrow \infty$, we have $1-P_s-P_L$ equal to $\sum P_i$ with $i \in \{BM-SC, GSN, RNC, UE\}$.

The dynamics of changing states per individual end-user can be modeled as a continuous-time Markov chain with the states mentioned above, where the transition rates are as depicted in Fig. 5.2.

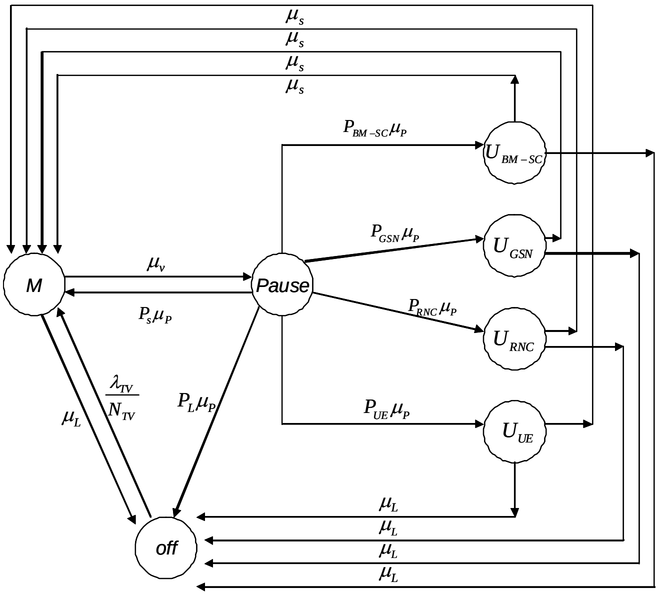


Figure 5.2: Continuous-time Markov chain for state transitions of a user who registered for a video streaming service.

We will now concentrate on analyzing this continuous-time Markov process.

1) The infinitesimal generator Q [32, pp.183] is explicitly given by Eq. (5.3), where the elements q_{xy} in Q at row x and column y reflect a change from state x towards state

$$Q=\begin{bmatrix} -(\mu_v+\mu_L) & \mu_v & 0 & 0 & 0 & 0 & \mu_L \\ P_s\mu_p & -\mu_p & P_{ROOT}\mu_p & P_{REX}\mu_p & P_{LEX}\mu_p & P_{RG}\mu_p & P_L\mu_p \\ \mu_s & 0 & -(\mu_s+\mu_L) & 0 & 0 & 0 & \mu_L \\ \mu_s & 0 & 0 & -(\mu_s+\mu_L) & 0 & 0 & \mu_L \\ \mu_s & 0 & 0 & 0 & -(\mu_s+\mu_L) & 0 & \mu_L \\ \mu_s & 0 & 0 & 0 & 0 & -(\mu_s+\mu_L) & \mu_L \\ \frac{\lambda_{TV}}{N_{TV}} & 0 & 0 & 0 & 0 & 0 & -\frac{\lambda_{TV}}{N_{TV}} \end{bmatrix} \tag{5.3}$$

y. We have $\sum_{y=1,x\neq y}^{7} q_{xy} = -q_{xx}$ [32, pp.181] and state 1 $= M$; state 2 $= Pause$; state 3 $= U_{BM-SC}$; state 4 $= U_{GSN}$; state 5 $= U_{RNC}$; state 6 $= U_{UE}$; state 7 $= off$.

2) We define a state vector $S(t) : \{M,\ Pause,\ U_{BM-SC},\ U_{GSN},\ U_{RNC},\ U_{UE},\ off\}$ with $\sum_{s=1}^{7} S_s(t) = 1$ (total law of probability), and we have $\lim_{t\to\infty} S(t) = \pi$.

Thus, the steady-state (row) vector π is a solution of

$$\pi Q = 0$$

In the following, the components of π are indexed as π_M, π_{Pause}, $\pi_{U_{BM-SC}}$, $\pi_{U_{GSN}}$, $\pi_{U_{RNC}}$, $\pi_{U_{UE}}$, and π_{off}.

5.4 Computation of the blocking probability

We present a simplified analysis in which user events are independent.

5.4.1 Definition of blocking probability $B(k)$

In steady state, $B(k)$ denotes *the probability that an individual user cannot get access to the service of his choice* when he requests the video channel k, or when he resumes it after pausing. We define $B_m(k)$ as the end-to-end (E2E) *multicast request blocking*, and $B_{u;i}$ as the E2E *unicast request blocking* if the end user retrieves the delayed video data from cache at level i after the pause. According to the total law of probability $\Pr[Blocking] = \sum_{s=1}^{7} \Pr[Blocking|state] \cdot \Pr[state]$, we have

$$\begin{aligned} B(k) &= [P_s B_m(k) + \sum_i P_i B_{u;i}] \cdot \pi_{Pause} \\ &\quad + B_m(k) \cdot (1 - \pi_{Pause}) \end{aligned} \tag{5.1}$$

where $i \in \{BM-SC, GSN, RNC, UE\}$.

$B_m(k)$ and $B_{u;i}$ will be computed in the following sections.

5.4.2 Computation of $B_m(k)$

To compute $B_m(k)$, we introduce two new probability functions $P(k)$ and $B_{Engset}(k)$. $P(k)$ is the probability that channel k is "on" and $B_{Engset}(k)$ is the probability that[1] the link L_{RNC} from RNC to BS is consumed by m multicast channels other than the requested channel k. Our computation of $B_{Engset}(k)$ is based on the $B_{Engset}(i)$ for IPTV deduced in the Appendix A.1, but with changing the symbol representing the TV channel index from i to k and also having some changes/adds on the following parameters: 1) the number of admitted channels left for multicast $m = \max\{1, \frac{L_{RNC}}{v} - N_{TV}(\pi_{U_{BM-SC}} + \pi_{U_{GSN}} + \pi_{U_{RNC}})\}$, and 2) $\lambda_k = \lambda_{k\_L_{RNC}}$, $u_k = u_{k\_L_{RNC}}$ and $\mu_k = \mu_{k\_L_{RNC}}$ (see Eqs. 5.3 and 5.7).

After knowing $B_{Engset}(k)$, we can get $B_m(k)$ using the following formula

$$B_m(k) = (1 - P(k))B_{Engset}(k) \tag{5.2}$$

$P(k)$: Similar to computing $P(i)$ when deducing the IPTV blocking probability in Section 4.4, here we can also compute $P(k)$, but with some changes on notations and deducing the arrival/departure rate. Disregarding possible blocking[2], channel k can be modeled as an $M/M/\infty$ queue (in steady-state), with infinite positions available in the queue to store all requests for channel k. Hence, the probability that channel k is "on", given that no requests are blocked, is equal to the probability that the $M/M/\infty$ queue [32, pp. 281] is not empty: $\Pr[N_{s;k} > 0] = 1 - \exp(-\rho_k)$, where $\rho_k = \lambda_k/u_k$. λ_k is the users' arrival rate in multicast channel k, and u_k is the number of users leaving from multicast channel k per second.

In order to assure content availability, a multicast user pausing the video is not considered as leaving. A user can only leave the multicast network in state $Pause$ and state M.

After the pause, resuming the channel k from the local cache UE is not considered as leaving from the network. Hence, the leaving rate from the multicast channel k after the $Pause$ is the sum of the leaving rates to all states (except to state U_{UE}), which is equal to $(1 - P_{UE})\mu_P$. The leaving rate from the multicast channel k after the state M is the sum of the leaving rates to state M and to state off (when switching to another channel and turning off the system), which is equal to $\mu_s + \mu_L$. Thus, the users' leaving rate u_k is equal to the number of channel k viewers at link L_{RNC} ($N_{TV}\alpha_k$) multiplied by the mean leaving rate of a multicast user $(\pi_M(\mu_s + \mu_L) + \pi_{Pause}(1 - P_{UE})\mu_P)$, where α_k represents the popularity of video channel k. In other words, an end-user has a probability of α_k to choose channel k. Similarly, we can also compute the users' arrival rate λ_k based on Fig. 5.2.

---

[1] The bandwidth bottleneck of the system is at L_{RNC}.

[2] If a user's request enters when channel k is "on", the request will be grouped into multicast. However, if a user's request enters when channel k is "off", it has to open a new channel, which might not be possible in case of insufficient available capacity.

The normalized user arrival rate in channel k and the normalized user leaving rate from channel k can be expressed as:

$$\begin{aligned}\frac{\lambda_k}{N_{TV}} &= (\pi_M + \pi_{U_{BM-SC}} + \pi_{U_{GSN}} + \pi_{U_{RNC}} + \pi_{U_{UE}}) \\ &\qquad (1\text{-}\alpha_k)\mu_s\alpha_k + \pi_{Pause}(1\text{-}\alpha_k)P_s\mu_p\alpha_k + \pi_{off}\frac{\lambda_{TV}}{N_{TV}}\alpha_k \\ \frac{u_k}{N_{TV}} &= \pi_M\alpha_k(\mu_s{+}\mu_L){+}\pi_{Pause}\alpha_k(1 - P_{UE})\mu_P \end{aligned} \tag{5.3}$$

Channel k can be either “off” or “on”. We therefore resort to a two-state Markov chain as illustrated in Fig. 4.2 in Section 4.4 to analyze its steady state.

5.4.3 Computation of $B_{u;i}$

$B_{u;i}$, with $i \in \{BM - SC, GSN, RNC, UE\}$, is defined as the probability that an arbitrary attempt to jump to the unicast state U_i is blocked because the amount of required bandwidth is not available at one of the links over the unicast connection. For instance, if a unicast user can retrieve data from cache $BM - SC$ without blocking ($B_{u;BM-SC} = 0$), that means the request cannot be blocked at all levels ($BM - SC$, GSN, and RNC). We assume that, if an end-user resumes the stream from his local cache C_{UE} after the pause, this attempt will never be blocked.

Based on the definition of $B_{u;i}$ and the assumption that the blocking at different levels are independent with each other, we have

$$\begin{aligned} &B_{u;BM-SC} = 1 - \prod_{i \in \{BM-SC, GSN, RNC\}} (1 - B_{u\_L_i}) \\ &B_{u;GSN} = 1 - \prod_{i \in \{GSN, RNC\}} (1 - B_{u\_L_i}) \\ &B_{u;RNC} = B_{u\_L_{RNC}} \\ &B_{u;UE} = 0 \end{aligned} \tag{5.4}$$

where

$$B_{u\_L_i} = \sum_{j=1}^{\min\{K, \lfloor L_i/v \rfloor\}} \pi_{j\_L_i} B_{u\_L_i\_j} \tag{5.5}$$

and $B_{u\_L_i}$ represents the mean probability that a unicast request is blocked at level i.

In (5.5), $\pi_{j\_L_i}$ represents the probability that j video streams are multicast at link L_i in steady state. K represents the amount of available video streams. $B_{u\_L_i\_j}$ represents the blocking probability of a unicast request at link L_i when j video streams are multicast at link L_i.

Our computation of $\pi_{j\_L_i}$ and $B_{u\_L_i\_j}$ will be given in the following.

Computation of $\pi_{j\_L_i}$

The probability that j positions are occupied by j MobileTV channels when there are K available video channels for this system can be deduced as follows:

According to [32, pp.18] and [31], the binomial probability generating function of $\pi_{j\_L_i}$ is

$$\varphi(z) = \sum_{j=0}^{\infty} \pi_{j\_L_i} z^j = \prod_{k=1}^{K} q_{k\_L_i} + p_{k\_L_i} z \tag{5.6}$$

where we always have $j < \min\{K, \lfloor L_i/v \rfloor\}$, $p_{k\_L_i} = 1 - e^{-\frac{\lambda_{k\_L_i}}{\mu_{k\_L_i}}}$ and $q_{k\_L_i} = 1 - p_{k\_L_i}$ for a particular link L_i, with $i \in \{BM - SC, GSN, RNC\}$.

$\lambda_{k\_L_i}$ represents the users' multicast requests arrival rate at video channel k (which is also the arrival rate of channel k) at link L_i. Next, $\mu_{k\_L_i}$ is the leaving rate of channel k from link L_i (which can be computed here via an $M/G/\infty$ model), and $u_{k\_L_i}$ is the channel k users' leaving rate at link L_i.

Beside the average number of registered users N_{TV} attaching to one BS, we use N_{RNC} and N_{GSN} to represent the number of branches connected to a RNC and to a GSN, respectively.

The users' multicast request arrival rate into link L_{GSN} is equal to the product of the users' multicast request arrival rate per link L_{RNC} and the number of L_{RNC} links connected to a RNC (see Fig. 5.1). Hence, we have $\lambda_{k\_L_{GSN}} = N_{RNC}\lambda_{k\_L_{RNC}}$. When computing $u_{k\_L_{GSN}}$ (the users' leaving rate from link L_{GSN}), state transitions from $Pause$ to U_{UE} and to U_{RNC} are not considered as leaving from link L_{GSN}. Similarly, we can also compute $\lambda_{k\_L_{BM-SC}}$ and $u_{k\_L_{BM-SC}}$.

According to Fig. 5.1 and 5.2, we have $\lambda_{k\_L_{RNC}} = \lambda_k$, $u_{k\_L_{RNC}} = u_k$ (see (5.3)); and both $\lambda_{k\_L_i}$ and $u_{k\_L_i}$ at other levels ($BM - SC$ and GSN) can be expressed as

$$\begin{aligned}
\lambda_{k\_L_{GSN}} &= N_{RNC}\lambda_{k\_L_{RNC}} \\
\lambda_{k\_L_{BM-SC}} &= N_{RNC}N_{GSN}\lambda_{k\_L_{RNC}} \\
u_{k\_L_{GSN}} &= N_{RNC}N_{TV}\pi_M\alpha_k(\mu_s + \mu_L) + N_{RNC}N_{TV} \\
&\quad \pi_{Pause}\alpha_k(1 - P_{UE} - P_{RNC})\mu_P \\
u_{k\_L_{BM-SC}} &= N_{RNC}N_{GSN}N_{TV}\pi_M\alpha_k(\mu_s+\mu_L)+N_{RNC}N_{GSN} \\
&\quad N_{TV}\pi_{Pause}\alpha_k(1 - P_{UE} - P_{RNC} - P_{GSN})\mu_P \\
\mu_{k\_L_i} &= \frac{\lambda_{k\_L_i}}{\exp(\frac{\lambda_{k\_L_i}}{u_{k\_L_i}} - 1)}
\end{aligned} \tag{5.7}$$

Computation of $B_{u\_L_i\_j}$

We use $L_{i\_j}^{(unicast)} = L_i - jv$ to represent the bandwidth for unicast when there are j multicast videos at level i, which corresponds to $\lfloor L_{i\_j}^{(unicast)}/v \rfloor = n_{i\_j}$ available unicast servers at level i. We assume that this unicast service at level i can be modeled as an $M/M/n$ queuing system. Based on [32, pp. 277] we have

$$\begin{aligned} B_{u\_L_i\_j} &= \Pr[N_s \geq n_{i\_j}] \\ &= \frac{\Pr[N_s = 0]}{n_{i\_j}!(1 - \frac{\lambda_i}{n_{i\_j}\beta_i})} \frac{\lambda_i^{n_{i\_j}}}{\beta_i^{n_{i\_j}}} \end{aligned} \tag{5.8}$$

where

$$\Pr[N_s = 0] = \frac{1}{\sum_{a=0}^{n_{i\_j}-1} \frac{\lambda_i^a}{a!\beta_i^a} + \frac{\lambda_i^{n_{i\_j}}}{n_{i\_j}!\beta_i^{n_{i\_j}}} \frac{1}{1 - \frac{\lambda_i}{n_{i\_j}!\beta_i}}}$$

λ_i is the arrival rate of unicast requests at L_i, and β_i is the leaving rate of unicast transmissions at link L_i.

For the arrival rate λ_{RNC}, we know that the unicast request has to pass through link L_{RNC} no matter from which higher-level ($BM - SC$, GSN, or RNC) the end user retrieves the delayed video after the pause. For the leaving rate β_{RNC}, we know that one unicast transmission at link L_{RNC} can be considered as leaving when a unicast stream from $BM - SC$ or GSN or RNC leaves. Similarly, λ_{GSN}, λ_{BM-SC}, β_{GSN} and β_{BM-SC} can also be computed.

$$\begin{aligned} \lambda_{RNC} &= N_{TV}\pi_{Pause}\mu_P(P_{BM-SC} + P_{GSN} + P_{RNC}) \\ \lambda_{GSN} &= N_{RNC}N_{TV}\pi_{Pause}\mu_P(P_{BM-SC} + P_{GSN}) \\ \lambda_{BM-SC} &= N_{RNC}N_{GSN}N_{TV}\pi_{Pause}\mu_P P_{BM-SC} \\ \beta_{RNC} &= N_{TV}(\pi_{U_{RNC}}+\pi_{U_{GSN}}+\pi_{U_{BM-SC}})(\mu_s+\mu_L) \\ \beta_{GSN} &= N_{RNC}N_{TV}(\pi_{U_{GSN}}+\pi_{U_{BM-SC}})(\mu_s+\mu_L) \\ \beta_{BM-SC} &= N_{RNC}N_{GSN}N_{TV}\pi_{U_{BM-SC}}(\mu_s+\mu_L) \end{aligned}$$

Finally, substituting the expression (5.6) for $\pi_{j\_L_i}$ and that of $B_{u\_L_i\_j}$ in (5.8) into (5.5) and further into (5.4), $B_{u;i}$ for different cases of i is found.

Furthermore, we can compute its mean unicast request blocking probability, using formula $E[B_{u;i}] = \sum\limits_i P_i B_{u;i}$ where $i \in \{BM - SC, GSN, RNC, UE\}$.

In the following section, we will apply our blocking model for this multicast plus unicast MobileTV service to a realistic case study.

5.5 Experiments

We applied our blocking model to the following realistic case study. We focus on a typical UMTS MobileTV network architecture for which we have assumed parameter values as shown in Table 5.1. The cache sizes are based on current smart phone specifications and reasonable capital expenditure for the network operator. The average pause time, which can be considered as the average call duration, is set to be 107.1 seconds based on the measurement study in [37].

Table 5.1: Parameter values in our case study

$v = 250$ Kbit/s (for 3.5G) [38]	$K = 12$
$N_{TV} = 600$	$C_{BM-SC} = 10$ TBytes
$N_{RNC} = 250$	$C_{GSN} = 800$ GBytes
$N_{GSN} = 3$	$C_{RNC} = 20$ GBytes
$\lambda_{TV}/N_{TV} = 1/600$	$C_{UE} = 1$ GBytes
$1/\mu_p = 107.1$ seconds	$L_{RNC} = 2$ Mbit/s
$1/\mu_s = 1/\mu_v = 1/\mu_L = 300$ s	$L_{GSN} = 70$ Mbit/s
$P_s = P_L = 1/100$	$L_{BM-SC} = 155$ Mbit/s

In a MobileTV system users do not view a TV channel for a long time. Therefore, the probability that the user wants to pause multiple times is low. Moreover, the channel popularity distribution α_k for the channels is obtained from a Dutch market survey. Channel 1 has a popularity of 15.1% and channel 23 of 0.2%. The first 23 number of TV channels cover in total 94.1%. The remaining channels uniformly share a popularity of 5.9%.

Fig. 5.3 plots the CDN overall blocking probability $B(k)$ as a function of the channel index k, for two different values of K and L_{RNC}.

The results show that the less popular TV channels (higher channel index) have a slightly lower probability to be blocked than the more popular channels, because the user arrival rate decreases slower than the user leaving rate, with increasing channel index k. Furthermore, Fig. 5.3 indicates that for $K = 12$, a 2.5 Mbit/s bandwidth between the RNC and the Base Station is enough to support a MobileTV service to the customers with the end-to-end blocking probability around 1% for all TV channel viewers. In addition, we can observe that the user's request is more likely to be blocked if there are more TV channels available (larger K), and this factor significantly affects the overall quality. This is illustrated in Fig. 5.4, which illustrates the blocking probability as a function of the number of available TV channels K for two different L_{RNC}.

When $K < 8$, at $L_{RNC} = 2$ Mbit/s, the value of the blocking probability is negligible, because K is smaller than the number of channels able to be transported simultaneously in the link L_{RNC} (i.e. $K \times v < L_{RNC}$). To maintain an acceptable user experience (with the overall blocking probability less than 1% for instance), a maximum of 8 Mobile TV channels can be provided in this case, however the maximum number of MobileTV channels can be 11 if L_{RNC} is dedicated to be 2.5 Mbit/s instead of 2 Mbit/s. If we want to maintain a total service availability of 99.99% or higher, 9 MobileTV channels

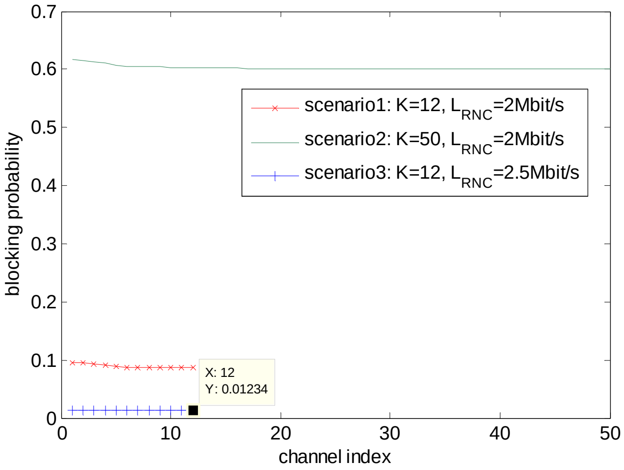


Figure 5.3: The overall blocking probablity as a function of the TV channel index (channel 1 is the most popular TV channel), with $C_{UE} = 1$ GBytes.

can be supported when $L_{RNC} = 2.5$ Mbit/s.

With same configuration and parameter settings as shown in Table 5.1, we also computed the mean unicast blocking probability $E[B_{u;i}]$ as a function of the local cache size at the end-user C_{UE}, ranging from 1 MByte to 1 GBytes. We found that the mean unicast blocking probability will decrease exponentially fast with the local cache size at the end user. Moreover, we observe that if an end-user wants to successfully resume the video stream after the pause with probability $> 99\%$, his local cache size C_{UE} should be at least 16 MByte, and a local cache size of ≥ 31 MByte is enough to guarantee the unicast service availability is larger than 99.99% after the pause. This easily matches current mobile phone specifications and the rest storage capacity (if the user has more than 31 MByte disk space at his mobile device) can support users' possible multiple pauses.

5.6 Summary and Conclusions

In this Chapter, a new blocking model for a hierarchical caching multimedia delivery system is derived, inspired by related work on home gateway caching in [5]. Different contributions to the blocking probability, namely multicast request blocking and unicast request blocking, have been analyzed separately.

Our model allows to compute how many MobileTV channels can be supported for the overall blocking probability not to exceed a certain threshold. Furthermore, we can compute the required local cache size to assure that an end-user can successfully resume

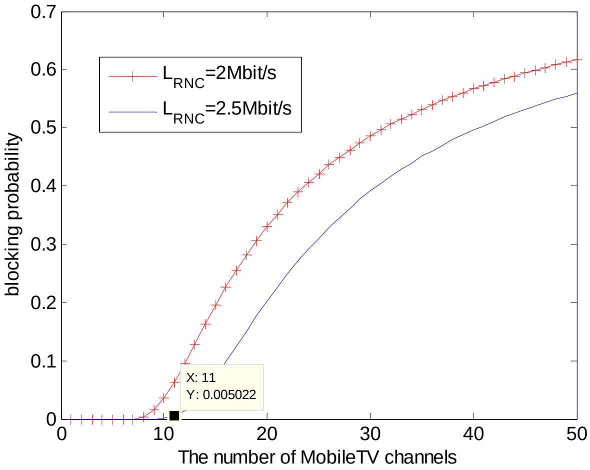


Figure 5.4: The overall blocking probablity as a function of the number of TV channels, with $C_{UE} = 1$ GBytes.

the streaming with a certain probability after the pause. Our results can be used not only to analyze the blocking in existing stream caching systems, but also to predict the system behavior. This may be helpful in the design of streaming systems.

Applying our model to a realistic MobileTV use case, we found that the number of available TV channels strongly affects the overall quality. Apparently, the less popular channels are still popular enough to end up with many unicast streams, and as such annihilate the efficiency advantage we hoped to create by multicasting the popular channels using a hierarchical caching architecture. When we increase the bandwidth between *RNC* and the *Base Station* with 25%, the maximum amount of channels that could be supported increased with 37.5% (with the cache sizes given in Table 5.1), and the advantage of having a CDN then becomes significant. The mean unicast blocking probability will decrease exponentially fast with the local cache size at the end user. To guarantee an availability of a unicast stream after pausing of more than 99.99%, for a network with the parameters of Table 5.1, the end user needs a local cache of only 31 MBytes or more at their device for this service. This requirement is easily matched by today's mobile phones.

Our model can also be adapted and dimensioned to scenarios with higher link capacity (such as foreseen for LTE networks) and correspondingly more (HD)TV channels can then be supported.

Part II

Video-on-Demand (VoD) Streaming Systems

Chapter 6

P2PVoD Model

The traditional Server-Client Video-on-Demand (VoD) Streaming system is similar to the unicast part of the CDN system mentioned in the previous Chapter. By removing the multicast function, the unicast request blocking in the CDN system is equivalent to the blocking which would appear in the Server-Client VoD Streaming system. However, in this part, we will particularly focus on another VoD Streaming system using Peer-to-Peer technology, called the P2PVoD system.

There has been a growing interest for Video-on-Demand (VoD) using Peer-to-Peer (P2P) technology. Unlike centralized solutions for VoD Streaming services, P2P technology lets the clients distribute video content among themselves. In this Chapter, we propose an analytical model for P2PVoD and we compare that model to a realistic P2PVoD simulator. With our model, parameters that affect the system performance can be observed, and the system stability can be investigated. Our model leads to design rules for achieving a good and stable system performance. This work as published in [39] is, to our knowledge, the first analytical work to model mesh-based P2PVoD.

6.1 Introduction

In P2PTV users can access the available TV channels to view the content that is being displayed at that particular point in time. In P2P Video-on-Demand (P2PVoD, e.g. Tribler [14]), users arrive at arbitrary points in time into the system to watch a video of their choice from its beginning. The question addressed in this Chapter is *how to use mesh-based P2P technology to provide P2PVoD services with good and stable performance.* To answer this question, we have developed a P2PVoD model[1].

The rest of this Chapter is organized as follows: In Section 6.2, related work is discussed. In Section 6.3, we develop our analytical model for P2PVoD. This model aims to present the number of downloaders and seeds watching a video i and the average

---

[1] When referring to P2PVoD, we mean mesh-based P2PVoD.

downloading speed at a peer as a function of time. After linearizing this analytical model in Section 6.4, we compare it with our simulation results in Section 6.5. Finally, we conclude the Chapter in Section 6.6.

6.2 Related work

Guo *et al.* [27] and Qiu *et al.* [40] model mesh-based P2P file sharing analytically and present a performance study combined with extensive measurements. In our work, we adopt an approach similar to [27] and [40] to explore the performance of mesh-based P2PVoD.

Kumar *et al.* [41] proposed a fluid model for mesh-based P2PTV, which has only one seed. Lu *et al.* [19] proposed a mesh-based model for P2PTV and compared its blocking to that of IPTV. However, the models in [41] and [19] are not applicable to P2PVoD.

Some research works (e.g., [42], [43]) target mesh-based P2PVoD, but only use simulations to analyze which kind of chunk-scheduling method can achieve the best performance. The proposed system by Chi et al. [44] was evaluated with the help of analytical models, but what they analyzed is tree-based P2PVoD [45], not mesh-based P2PVoD.

Prior to our work, an analytical model of mesh-based P2PVoD seemed missing.

6.3 A general fluid model for P2PVoD

Before modeling and analyzing P2PVoD, its basic mechanism and characteristics are addressed in order to better understand the behavior of the P2PVoD system. The content of P2PVoD is a video lasting a fixed amount of time. The video in P2PVoD can be divided into chunks. Each video has a unique ID, e.g. i. All chunks of video i can be found at the seeds. The distribution of video i starts with one initial seed, the original video i content provider.

A peer arrives at arbitrary points in time into the system to watch video i from its beginning. In our model, each P2PVoD peer stores the video i's content on his computer until he stops viewing this video. Thus, once a peer obtains a chunk, he makes the chunk available for downloading by other peers until he leaves.

We refer to the peer who is still downloading as “downloader” and refer to the peer who already finished the download, but is still viewing the video as “seed.” A peer joins the system as a downloader and contacts other peers[2] in order to download chunks of

---

[2] A new peer will choose some other peers who are also watching this video to form a neighbor group. Within this group, he can download what he needs from other peers based on the chunk availability information.

video i. After a prebuffering period, the peer starts the playback and from then on the video content is displayed, while at the same time the near-future video content is downloaded. After the peer has finished downloading the whole video file, he will become a seed until he departs.

In our P2PVoD model, there are two kinds of peer departures. One of them is the random departure and the other is the definite departure. A downloader may leave the network randomly at rate θ_i before the download is completed (e.g., when he feels that the video is boring). Even though a peer becomes a seed, he may still be viewing the video. Hence, a seed may leave the network randomly at a rate of θ_i before the video playback has completed. Nevertheless, the seed will definitely leave after he has viewed the video, with definite seed leaving rate $\gamma_i(t)$ (we can consider $1/\gamma_i(t)$ to be the seed serving time, see Fig. 6.1).

A peer generally obtains video chunks in playback order. Hence, a peer has to download the near-future video chunks as high priority within a downloading time limit.

In our P2PVoD network, beside seeds who will definitely upload data, a downloader who has not finished downloading yet can also upload data to other downloaders. The number of downloaders at t is $x_i(t)$ and the number of seeds at t is $y_i(t)$. A downloader has probability $\eta_i(t)$ to be used for sharing his content with others. Based on our simulations, the value of $\eta_i(t)$ is approaching 1 except for the first few seconds of the system. Hence, we can simplify the model by setting $\eta_i(t) = 1$.

We list the symbols, which will be used in our P2PVoD model, in Table 6.1.

Table 6.1: Symbols

v: Video playback rate (Mbit/s).
L_i: The length (in seconds) of video i.
$\lambda_i(t)$: Peers' arrival rate for video i at time t.
θ_i: Peer's random leaving rate from video i.
$\gamma_i(t)$: Seed's definite leaving rate. $\gamma_i(t) = \frac{1}{\text{seed serving time}}$.
$x_i(t)$: No. of downloaders in the video i system at t.
$y_i(t)$: No. of seeds in the video i system at t.
bw_{os}: The upload rate of the original source provider.
bw_{up}: Avg. upload rate at a peer for video delivery.
bw_{down}: Max download rate of a peer for video delivery.
$u_i(t)$: Avg. download rate of a peer at t in video i system.
$T_i(t)$: Time a peer needs for downloading the video i at t.
τ_i: The time interval from the time that video i appeared to the time that the first seed appears in the system.

6.3.1 Model description

Our analysis of mesh-based P2PVoD can be considered to be a worst-case study. We do not consider any extra complex strategies (like peer selection, incentive management, failure management, chunk scheduling, etc.). We use a fluid model to compute the time-dependent average number of downloaders and seeds in system i. Hence, we do not compute any fluctuations around the average.

We will show that our general P2PVoD model leads to a non-linear system. Consequently, we shall analyze which factors cause this non-linearity such that we might redesign our P2PVoD system to become linear in all conditions.

In the following, we introduce ordinary differential equations to express our fluid model in general.

The total uploading rate of the system can be expressed as $\min\{bw_{down}x_i(t), bw_{up}(x_i(t)+y_i(t)) + bw_{os}\}$, where bw_{down} is the download rate upperbound for video delivery; bw_{up} is the average upload rate at a peer for video delivery; bw_{os} is the upload rate of the original source provider (we assume only one original source provider for one video); $x_i(t)$ and $y_i(t)$ respectively represent the number of downloaders and seeds for video i at time t. If there is enough downloading bandwidth, the total uploading rate of the system reduces to $bw_{up}(x_i(t) + y_i(t)) + bw_{os}$. At time t, the overall downloading rate related to video i is equal to the overall uploading rate related to video i: $u_i(t)x_i(t) = \min\{bw_{down}x_i(t), bw_{up}(x_i(t) + y_i(t)) + bw_{os}\}$. Hence, we express the average download rate $u_i(t)$ as

$$u_i(t) = \frac{\min\{bw_{down}x_i(t), bw_{up}(x_i(t) + y_i(t)) + bw_{os}\}}{x_i(t)} \tag{6.1}$$

In order to analyze the system performance for video i, we need to calculate $u_i(t)$. In order to obtain $u_i(t)$, we should first get the values of $x_i(t)$ and $y_i(t)$, which can be obtained by solving Eqs. (6.2) to (6.5), explained below.

Each peer joins the P2PVoD system as a downloader. After finishing the download, a downloader will become a seed. At time t, the total downloading rate $u_i(t)x_i(t)$ (Mbit/s) divided by the length of the video $L_i v$ (Mbits) can be considered as the rate at which downloaders become seeds. Continuing with this idea, the downloaders' generating rate $\frac{dx_i(t)}{dt}$ should be equal to the downloaders' arrival rate $\lambda_i(t)$ minus the downloaders' leaving rate $\theta_i x_i(t)$ and minus the rate of downloaders becoming seeds $\frac{u_i(t)}{L_i v}x_i(t)$.

$$\frac{dx_i(t)}{dt} = \lambda_i(t) - \theta_i x_i(t) - \frac{\min\{bw_{down}x_i(t), bw_{up}(x_i(t)+y_i(t))+bw_{os}\}}{L_i v} \tag{6.2}$$

In our fluid model, the peer arrival process can be any kind of process. For simplicity, we consider the average arrival rate as a constant value, $\lambda_i(t) = \lambda_i$.

The seeds' generating rate $\frac{dy_i(t)}{dt}$ should be equal to the rate of downloaders becoming seeds $\frac{u_i(t)}{L_i v}x_i(t)$ minus the seeds' leaving rate $(\theta_i + \gamma_i(t))y_i(t)$. Thus,

$$\frac{dy_i(t)}{dt} = \frac{\min\{bw_{down}x_i(t), bw_{up}(x_i(t)+y_i(t))+bw_{os}\}}{L_i v} - (\theta_i + \gamma_i(t))y_i(t), \tag{6.3}$$

Eqs. (6.2) and (6.3) show that there still is one unknown variable: the seed definite departure rate $\gamma_i(t)$. We deduce $\gamma_i(t)$ below.

If we make a peer (seed) depart as soon as the display ends[3], our P2PVoD system is possibly non-linear, where the seed's definite departure rate $\gamma_i(t)$ depends on $x_i(t)$ and $y_i(t)$.

We obtained the equations of $\gamma_i(t)$ and $T_i(t)$ based on Fig. 6.1. When a peer finishes downloading the video, it will become a seed until the video finishes and the peer departs. If a peer downloads the data very fast (high downloading speed), this peer will have a longer seed service time. The service time of a seed at time t, regardless of the peer arbitrary departures during the viewing, is equal to $1/\gamma_i(t) = L_i + B_u - T_i(t)$. For a given peer, the downloading time $T_i(t)$ times the downloading rate $u_i(t)$ is equal to the video size $L_i v$.

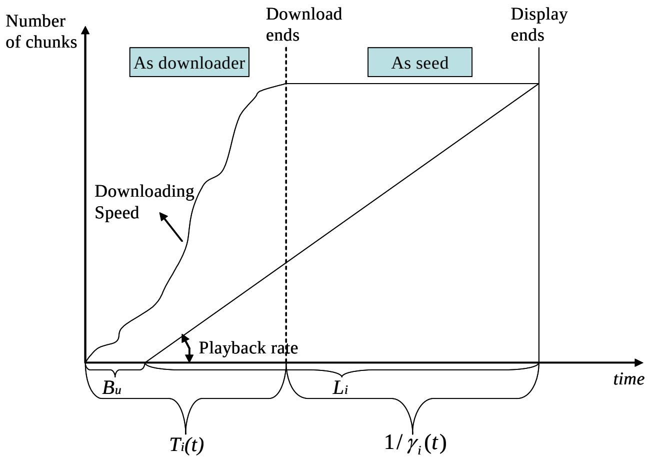


Figure 6.1: A peer U changes from a downloader status to a seed status.

When assuming that a peer definitely departs as soon as the display ends, we obtain

$$\gamma_i(t) = \frac{1}{L_i + B_u - T_i(t)}, \tag{6.4}$$

---

[3] The seed serving time depends on the average download time, which is determined by the number of downloaders and seeds in the system.

where the downloading time equals

$$T_i(t) = \max\left\{\frac{L_i v}{bw_{down}}, \frac{x_i(t)L_i v}{bw_{up}(x_i(t)+y_i(t))+bw_{os}}\right\}, \tag{6.5}$$

Eqs. (6.4) and (6.5) indicate that $T_i(t)$ may depend on $x_i(t)$ and $y_i(t)$, while $\gamma_i(t)$ depends on $T_i(t)$; thus $\gamma_i(t)$ depends on $x_i(t)$ and $y_i(t)$. Eq. (6.3) shows that $y_i(t)$ depends on $\gamma_i(t)$. Thus, when the download capacity is large, the seed definite leaving rate $\gamma_i(t)$ depends on $x_i(t)$ and $y_i(t)$, making $x_i(t)$ and $y_i(t)$ non-linear.

After having introduced the general idea of how to use differential equations to model a P2PVoD system, we are going to analyze four phases of the system: Start-up phase, Seed Appearance (SA) phase, Seed Departure (SD) phase, and Steady-state.

6.3.2 Start-up phase ($0 \leq t < \tau_i$)

The system starts with one original source provider. Thus, the initial number of downloaders is $x_i(0) = 0$. We use $y_i(t)$ to express the number of seeds in the system at time t. Here, $y_i(t)$ excludes the original source provider and $y_i(t) = 0$ when $0 \leq t < \tau_i$. The number of seeds in the system stays zero until a peer finishes downloading the whole video file and becomes the first seed. At the very beginning phase of the video i system, $0 \leq t < \tau_i$, the first downloader is able to download video content with the download rate of $u_i(t)$ and $\tau_i = L_i v / u_i(t)$.

We define the Start-up phase as the time interval between the availability of the video and the appearance of the first seed:

$$0 \leq t < \tau_i \begin{cases} \frac{dx_i(t)}{dt} = \lambda_i - \theta_i x_i(t), \\ y_i(t) = 0, \\ u_i(t) = \frac{bw_{up}x_i(t)+bw_{os}}{x_i(t)} \approx bw_{up} \end{cases}$$

6.3.3 Seed Appearance phase ($\tau_i \leq t < L_i$)

We assume that all peers are able to finish the download before the display ends. We define the SA phase as the time interval between the appearance of the first seed and the definite departure of the first seed. In this phase, no definite departures of seeds occur and no videos are released.

Thus, Eqs. (6.2) and (6.3) are simplified with $\gamma_i(t)$=0.

6.3.4 Seed Departure phase ($t \geq L_i$)

After the SA phase, the system enters the so-called SD phase ($t \geq L_i$). We define the SD phase as the time interval between the departure of the first seed to the start of steady-state.

In this phase, the seed's definite leaving rate $\gamma_i(t)$ is not equal to zero anymore. Eqs. (6.4) and (6.5) show that the value of $\gamma_i(t)$ depends on $T_i(t)$, which has different expressions under different conditions and which depends on $x_i(t)$ and $y_i(t)$ when $\frac{L_i v}{bw_{down}} < \frac{x_i(t)L_i v}{bw_{up}(x_i(t)+y_i(t))+bw_{os}}$.

The conditions referred to above will be analyzed in Section 6.3.5 on the steady-state. However, the conditions deduced for the steady-state can also be used to analyze the SD phase, when we consider the SD phase (varying with time t) as built-up by many quasi-steady states, each of which is lasting a unit (e.g., 1 second) of time. We can analyze the SD phase at different time points varying around different equilibrium points $\{\bar{x}_i, \bar{y}_i\}$ deduced in Section 6.3.5 with different values of γ_i.

In Section 6.5.1, we have used Matlab to compute the results of $x_i(t)$ and $y_i(t)$ based on the Eqs. (6.2) to (6.5) and the various conditions presented in the following section. The conditions relate to the value of γ_i and the condition that either the upload bandwidth or the download bandwidth is the constraint[4].

6.3.5 Steady-state

Analogous to the steady-state analysis for P2P file sharing in [40], we can find expressions for our P2PVoD model.

In steady-state, nothing varies with time t. Hence, Eqs. (6.2) and (6.3) become

$$\lambda_i - \theta_i \bar{x}_i - \min\{\frac{bw_{down}\bar{x}_i}{L_i v}, \frac{bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os}}{L_i v}\} = 0$$

$$\min\{\frac{bw_{down}\bar{x}_i}{L_i v}, \frac{bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os}}{L_i v}\} - (\theta_i + \gamma_i)\bar{y}_i = 0$$

where $\bar{x}_i = \lim\limits_{t\to\infty} x_i(t)$ and $\bar{y}_i = \lim\limits_{t\to\infty} y_i(t)$ are the equilibrium values of $x_i(t)$ and $y_i(t)$.

1) We can solve these equations if $\frac{bw_{down}\bar{x}_i}{L_i v} \le \frac{bw_{up}(\bar{x}_i+\bar{y}_i)+bw_{os}}{L_i v}$ (the download bandwidth is the constraint) as

$$\bar{x}_i = \frac{\lambda_i}{\frac{bw_{down}}{L_i v} + \theta_i} \tag{6.6}$$

$$\bar{y}_i = \frac{\lambda_i}{(\gamma_i + \theta_i)(1 + \frac{\theta_i L_i v}{bw_{down}})} \tag{6.7}$$

where $\gamma_i = \frac{1}{L_i + B_u - (L_i v / bw_{down})}$.

With the expressions of $\bar{x}_i$ and $\bar{y}_i$, the assumption that $\frac{bw_{down}\bar{x}_i}{L_i v} \le \frac{bw_{up}(\bar{x}_i+\bar{y}_i)+bw_{os}}{L_i v}$ amounts to $\frac{L_i v}{bw_{down}} \ge \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$, which is equivalent to

---

[4] Here, the download bandwidth is the constraint stands for $bw_{down}x_i(t) \le bw_{up}(x_i(t) + y_i(t)) + bw_{os}$, *not* $bw_{down} \le bw_{up}$. Accordingly, the upload bandwidth is the constraint means $bw_{down}x_i(t) \ge bw_{up}(x_i(t) + y_i(t)) + bw_{os}$, *not* $bw_{down} \ge bw_{up}$.

$$\gamma_i \leq \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i \tag{6.8}$$

Thus, when $\gamma_i \leq \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$, we use (6.6) and (6.7) to express the number of downloaders and seeds in steady state. And in this case, the download bandwidth per peer is fully used in steady state, with $u_i = bw_{down}$.

2) On the other hand, if $\frac{bw_{down}\bar{x}_i}{L_i v} > \frac{bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os}}{L_i v}$ (the upload bandwidth is the constraint), we obtain

$$\bar{x}_i = \frac{\lambda_i \theta_i L_i v + \lambda_i \gamma_i L_i v - bw_{up}\lambda_i - bw_{os}\theta_i - bw_{os}\gamma_i}{S} \tag{6.9}$$

$$\bar{y}_i = \frac{\lambda_i bw_{up} + bw_{os}\theta_i}{S} \tag{6.10}$$

where $S = bw_{up}\gamma_i + \theta_i \gamma_i L_i v + \theta_i^2 L_i v$, $\gamma_i = \frac{1}{L_i + B_u - (L_i v / u_i)}$ and $u_i = \frac{bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os}}{\bar{x}_i}$.

With the expressions of $\bar{x}_i$ and $\bar{y}_i$ above, the assumption that $\frac{bw_{down}\bar{x}_i}{L_i v} > \frac{bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os}}{L_i v}$ can also be expressed as $0 < \frac{L_i v}{bw_{down}} < \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i + \theta_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$.

Eqs. (6.9) and (6.10) should be used only when $\lambda_i L_i v > \frac{\lambda_i bw_{up}}{\gamma_i + \theta_i} + bw_{os}$ and if $\frac{L_i v}{bw_{down}} < \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i + \theta_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$, which is equivalent to

$$\gamma_i > \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$$

Thus, when $\gamma_i > \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$, we use (6.9) and (6.10) to express the number of downloaders and seeds in steady state.

This analysis shows that the characteristics of the seed's definite departure rate $\gamma_i(t)$ directly determine whether the system equations are linear or not. The downloading time of the whole file $T_i(t) = \frac{L_i v}{bw_{down}}$ in (6.5) leads to $\gamma_i(t)$, which is independent of $x_i(t)$ and $y_i(t)$, resulting in linear system equations; while $T_i(t) = \frac{x_i(t) L_i v}{bw_{up}(x_i(t) + y_i(t)) + bw_{os}}$ in (6.5) leads to non-linear system equations. An example of the issues brought by non-linear equations can refer to Section 6.5.1. In the following section, we will redesign our P2PVoD model, such that it becomes linear in all conditions. If the conditions (e.g., the bandwidth of end users or the user behavior) cannot be controlled, it is important to linearize the model to achieve a stable system performance *in all conditions*.

For the Start-up phase and Seed Appearance phase, there is no effect of $\gamma_i(t)$, because $\gamma_i(t) = 0$. These two phases always lead to linear equations. However, the Seed Departure phase and the steady state need linearization, because $\gamma_i(t)$ may depend on $x_i(t)$ and $y_i(t)$ in these two phases.

6.4 Linearization of the P2PVoD model

In order to obtain a linear system under all conditions, the seed serving time $\frac{1}{\gamma_i(t)}$ must be constant (i.e., a peer departs a fixed amount of time after his download finishes[5]). In other words, regardless of whether peers can download fast enough, we will let peers finish downloading the whole video and continue to share it for a while. We can design P2PVoD applications to obey this rule. But, how long should the video content be kept and shared after the download ends (what threshold should the value of $\frac{1}{\gamma_i}$ satisfy)? Intuitively, the longer peers' content is shared, the better for the whole system, but the more hard disk space and upload bandwidth resources are used at each peer.

Hence, we want the value of $\frac{1}{\gamma_i}$ to be as small as possible, but with the prerequisite that it is large enough to compensate the performance of the whole system. In the following, we analyze the system behavior in different conditions to deduce the best value for the parameter γ_i.

The linear differential equations can be expressed as

$$\frac{dZ(t)}{dt} = A_j Z(t) + b_j, \quad j = 1, 2 \tag{6.11}$$

where $Z(t) = \begin{bmatrix} x_i(t) \\ y_i(t) \end{bmatrix}$.

There are two possibilities ($j = 1$ and $j = 2$) based on the conditions deduced in Section 6.3.5:

1. Case $j = 1$, where $\gamma_i \leq \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$:

   Whether the system reaches a steady state is only determined by bw_{down}, even when the download bandwidth is large:

   $A_1 = \begin{bmatrix} -(\theta_i + \frac{bw_{down}}{L_i v}) & 0 \\ \frac{bw_{down}}{L_i v} & -(\theta_i + \gamma_i) \end{bmatrix}$ and
   $b_1 = \begin{bmatrix} \lambda_i \\ 0 \end{bmatrix}$

   The eigenvalues [46] of A_1 are $\mu_1 = -(\theta_i + \frac{bw_{down}}{L_i v})$ and $\mu_2 = -(\theta_i + \gamma_i)$, which are both negative.

   Since both eigenvalues are negative, we have a stable system that converges exponentially fast in t to the steady state.

   (a) Steady state ($t \to \infty$):

---

[5] It will be no problem for a P2PVoD developer to achieve this (just make the video content stored at a seed stop being shared, even when he is still viewing the video).

$\begin{bmatrix} \bar{x}_i \\ \bar{y}_i \end{bmatrix} = \frac{\lambda_i}{(\theta_i + \frac{bw_{down}}{L_i v})(\theta_i + \gamma_i)} \begin{bmatrix} (\theta_i + \gamma_i) \\ \frac{bw_{down}}{L_i v} \end{bmatrix}$, which gives the same[6] expressions as (6.6) and (6.7).

(b) Sensitivity of eigenvalues:
A larger value of the normalized download upperbound rate $\frac{bw_{down}}{L_i v}$ and a larger value of γ_i, under the condition that $\gamma_i \leq \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$, will cause the eigenvalues to have larger negative values, which on its turn will cause the number of downloaders and seeds $x_i(t)$ and $y_i(t)$ to reach a steady-state faster.

2. Case $j = 2$, where $\gamma_i > \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$:

$A_2 = \begin{bmatrix} -(\theta_i + \frac{bw_{up}}{L_i v}) & -\frac{bw_{up}}{L_i v} \\ \frac{bw_{up}}{L_i v} & \frac{bw_{up}}{L_i v} - (\theta_i + \gamma_i) \end{bmatrix}$ and $b_2 = \begin{bmatrix} \lambda_i - \frac{bw_{os}}{L_i v} \\ \frac{bw_{os}}{L_i v} \end{bmatrix}$

The eigenvalues of A_2 are

$\mu_1 = \frac{-(2\theta_i + \gamma_i) + \sqrt{\gamma_i^2 - 4\frac{bw_{up}}{L_i v}\gamma_i}}{2}$ and $\mu_2 = \frac{-(2\theta_i + \gamma_i) - \sqrt{\gamma_i^2 - 4\frac{bw_{up}}{L_i v}\gamma_i}}{2}$. Although complex, both eigenvalues always have negative real parts, which again shows that the system is stable.

(a) Steady state ($t \to \infty$):

$\begin{bmatrix} \bar{x}_i \\ \bar{y}_i \end{bmatrix} = \frac{1}{(\theta_i + \frac{bw_{up}}{L_i v})(\theta_i + \gamma_i - \frac{bw_{up}}{L_i v}) + (\frac{bw_{up}}{L_i v})^2}$
$\begin{bmatrix} (\theta_i + \gamma_i - \frac{bw_{up}}{L_i v})(\lambda_i - \frac{bw_{os}}{L_i v}) - \frac{bw_{up} bw_{os}}{(L_i v)^2} \\ \frac{bw_{up}}{L_i v}(\lambda_i - \frac{bw_{os}}{L_i v}) + \frac{bw_{os}}{L_i v}(\theta_i + \frac{bw_{up}}{L_i v}) \end{bmatrix}$, which is the same as Eqs. (6.9) and (6.10).

(b) Sensitivity of eigenvalues:
Given $\gamma_i > \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$, a larger value of γ_i will make the system reach the steady state faster.

On the other hand, if the download bandwidth is the constraint, the model equation changes to be the same as for case 1.

Based on the formulae deduced above, we can examine the effect of some parameters on the system behavior:

1) *What is the effect of* bw_{os}*?*

---

[6] Under the condition that γ_i is independent of $\bar{x}_i$ and $\bar{y}_i$.

The upload bandwidth of the original source provider bw_{os} does not affect the steady-state at all in case $j = 1$. In case $j = 2$, the larger the bw_{os}, the less downloaders and the more seeds in steady state (see Eqs. (6.9) and (6.10)), which leads to a larger average download rate $u_i(t)$ according to Eq. (6.1). Hence, a larger bw_{os} is helpful for the system performance when the average seed serving time is small and the peers' upload bandwidth is limited.

2) *What is the effect of bw_{up}?*

Assuming all peers are able to finish the download before the display ends, if we change the average upload bandwidth of the normal peers bw_{up} in case $j = 1$, the number of downloaders and seeds in steady state will not change. In case $j = 2$, the larger the bw_{up}, the less downloaders and the more seeds in steady state if $\gamma_i < \lambda_i$ (see Eqs. (6.9) and (6.10)). Hence, a larger bw_{up} is helpful for the system performance when the average seed serving time is small and the peers' upload bandwidth is limited. When peers cannot finish the download before the planned display ending time, the effect of bw_{up} will be discussed in Section 7.1.

3) *What is the effect of the peer arrival rate λ_i?*

Assuming $\theta_i = 0$ and ignoring the comparably small $\frac{bw_{os}}{L_i v}$, if we double the peer arrival rate λ_i, we can find that the number of downloaders and seeds in steady state will double in all conditions. Hence, we could normalize our system equations in steady state, with the number of peers divided by λ_i.

4) *What is the effect of the seed serving time $1/\gamma_i$?*

If we consider the two conditions individually, a larger value of the seed definite departure rate γ_i will make the system reach the steady state faster. However, if we consider the two conditions simultaneously, our best choice is to set the value of γ_i close to, but not exceeding, $\frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i$. Hence, a peer should stay at least $1/(\frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}} - \theta_i)$ seconds after the download finishes. As such we will have a stable system that not only converges faster to the steady-state, but also contains a larger number of seeds. For further analyzing the effect of this factor, detailed experiment results will be shown in Section 6.5.2.

6.5 Experiments

In this section, we compare our analytical results with our simulation results, under the same conditions.

For the computational part, we can feed the parameters into our fluid model and solve the ordinary differential equations with Matlab. We set the parameters as shown in Table 6.2.

For the simulation part, we have set up a discrete-event simulator. The transmission of chunks in the simulator is discrete, as opposed to our fluid model. A policy is therefore

Table 6.2: Value of parameters in experiments

$v = 0.5Mbit/s$ (for a video with TV quality),
$bw_{up} = 0.9Mbit/s$,
$bw_{os} = 4Mbit/s$,
$bw_{down} = 10Mbit/s$,
$\lambda_i = 1$, $B_u = 10\,\text{sec}$,
$L_i = 5\,\text{min}$ (e.g., a short YouTube-like video clip).

needed to determine for each peer which chunk it will download from which neighbor. The chunks are transferred from peer i to a neighbor j as follows. Peer i keeps its neighbors informed about the chunks it has finished downloading. Peer j can request these chunks and each request is granted by appending the chunk to the send buffer of peer i. To increase the chunk availability, while maintaining VoD behavior, we let peers download chunks at random within a window of B_u seconds starting from the first chunk that has not yet been downloaded. Since playback starts B_u seconds after the download starts, a peer typically notices no difference due to this change in policy.

Each simulation starts with one initial seed, and the peers arrive according to a Poisson process. The different departure processes will be explained in Section 6.5.1 and 6.5.2 individually. The simulation results are averaged over 20 runs. We found the average bandwidth utilization rate of a peer (upload rate/upload capacity) to be equal to 80% on average, while the bandwidth utilization rate of the original source provider is nearly 1. Hence, in order to be consistent with the settings in our fluid model, we set the original source provider's upload capacity to 4 Mbit/s, and a normal peer's upload capacity to $\frac{0.9}{80\%} = 1.1$ Mbit/s. The other parameters are equal to the fluid model.

Figs. 6.2, 6.3 and 6.4 show the number of downloaders and seeds, as well as the average download rate as a function of time t. We can imagine that the more popular video i is, the more downloaders and seeds there will be in the system.

6.5.1 General non-linear system

Peers arrive at a rate of λ_i and depart only when the playback is finished ($\theta_i = 0$). Each peer stores the video i's content until this video ends displaying ($\gamma_i(t)$ depends on $x_i(t)$ and $y_i(t)$).

Fig. 6.2 illustrates that, since the video was made available, the number of seeds increased until it reached a steady state, while the number of downloaders increased at first and then decreased suddenly into a steady state. The average downloading rate at a peer is small in the start-up phase and reaches its maximum in the steady state.

Comparing the analytical results and simulation results, they closely match except for the time of the start-up phase and SA phase. That is because we let the average download rate in start-up phase $u_i(0 \leq t \leq \tau_i)$ be roughly equal to bw_{up} in our math-

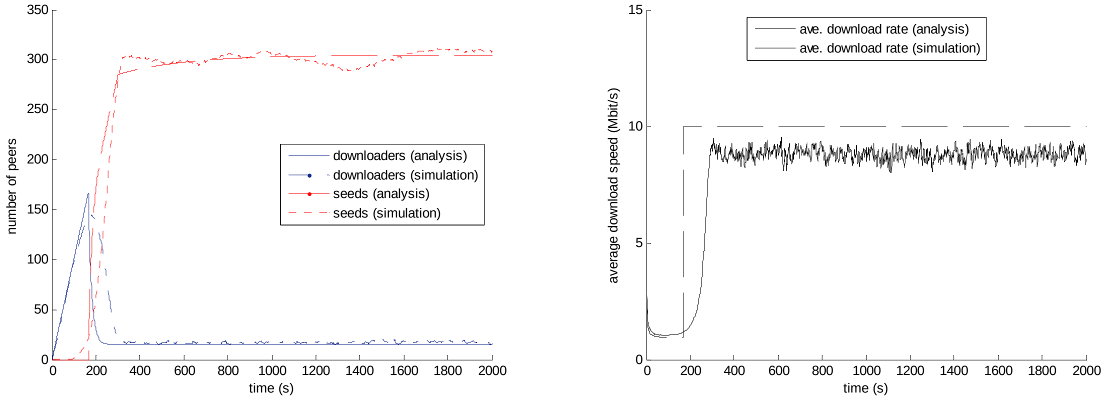


Figure 6.2: The number of downloaders $x_i(t)$ and seeds $y_i(t)$ (left) and the average download speed per peer $u_i(t)$ (right) as a function of time in a non-linear video i system.

ematical model, while in the real case (in simulation), $u_i(0 \leq t \leq \tau_i) = bw_{up} + \frac{bw_{os}}{\lambda_i(t)t} > bw_{up}$, which leads to a smaller $\tau_i = L_i v / u_i(t)$. Correspondingly, the peak number of downloaders in the start-up phase in the simulation was a little bit smaller than the one in the mathematical model. Nevertheless, the peak number of downloaders and seeds in our simulations, as well as the steady states, still closely match the analytical results. Our fluid model can thus be used to predict these numbers, which can then be used in the design of P2PVoD algorithms. Based on the knowledge of the number of downloaders and seeds in both start-up phase and steady state, the service operator can predict the average download rate at the end user to see whether it is enough for a smooth display. If it is not, maybe some helpers need to be added into the system (see Chapter 7).

Until seeds actually depart (before SD phase), the number of seeds plus the number of downloaders can be derived from the arrival rate and has to be equal in both the analysis and simulations. Then, in the SA phase, the differences between the results for the seeds and the downloaders are equal, which can be observed in Fig. 6.2.

The differences in download speed are due to the same reason. Our simulation tracks the average download speed with a history of 10 seconds, and combined with a constant arrival of peers with an initial download speed of 0. Thus, the average download speed is lower in the simulation than is predicted by the fluid model.

Other differences between the analytical and simulation results can be caused by the possible peer correlation and the flexible and stochastic P2P network under simulation. This includes slower, but more fluent, transitions between states when compared to the fluid model.

With our settings above, this non-linear system seems to perform very well. However, non-linear systems might be unstable. For instance, if too many ADSL peers are watching this video i (if we change $bw_{up} = 0.9Mbit/s$ to $bw_{up} = 0.4Mbit/s$), on average there will be no peers able to finish the download before the display ends. In this case, there will be no seeds and the average download rate will be always smaller than the playback rate, which causes blocking everywhere in this P2PVoD system (including all 4 phases). However, the linearized system can mitigate the problem and make seeds appear in steady state, which further leads to the fact that the average download rate can be large enough in steady state at least. The linearized system however still has some limitations which will be specified in Section 7.1.

6.5.2 Linearized system

We use the same values for our parameters, except for the value of γ_i. Because $\frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os}} \approx 0.00616$, we set:

(1) $\gamma_i = 0.006$ (which can be considered the threshold); Each user keeps his stored video for $\frac{1}{0.006} \approx 167$ seconds after he finishes the download, no matter how fast he downloaded it (see Fig. 6.3).

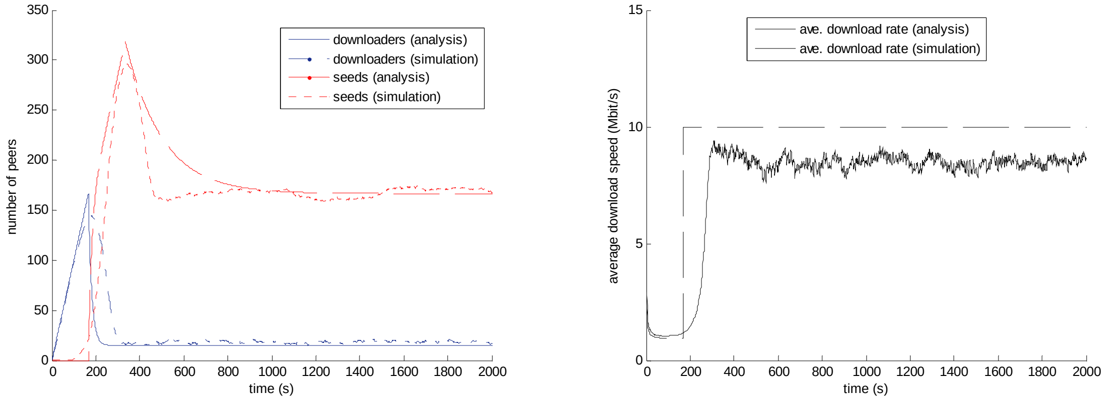


Figure 6.3: The number of downloaders $x_i(t)$ and seeds $y_i(t)$ (left) and the average download speed per peer $u_i(t)$ (right) as a function of time in a linearized system with γ_i=0.006.

We can see that this linear case with $\gamma_i < \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os}}$ has a similar system performance as its non-linear counterpart, because the average download rate at a peer is similar, even though the number of seeds in this case is smaller in steady state. The difference between the analytical results and the simulation results are similar to those in the non-linear system.

Furthermore, once $\gamma_i < \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os}}$, the average download rate will always be maximized. Therefore keeping the video at least $\frac{\lambda_i L_i v - bw_{os}}{\lambda_i bw_{up}}$ seconds after becoming a seed is indeed helpful to lead to a better system performance. Thus, the threshold of $\frac{\lambda_i L_i v - bw_{os}}{\lambda_i bw_{up}}$ is meaningful for P2PVoD application developers.

(2) $\gamma_i = 0.008$; Each user keeps his stored video for $\frac{1}{0.008} = 125$ seconds after he finishes the download. The situation of the linear system in this case is shown in Fig. 6.4.

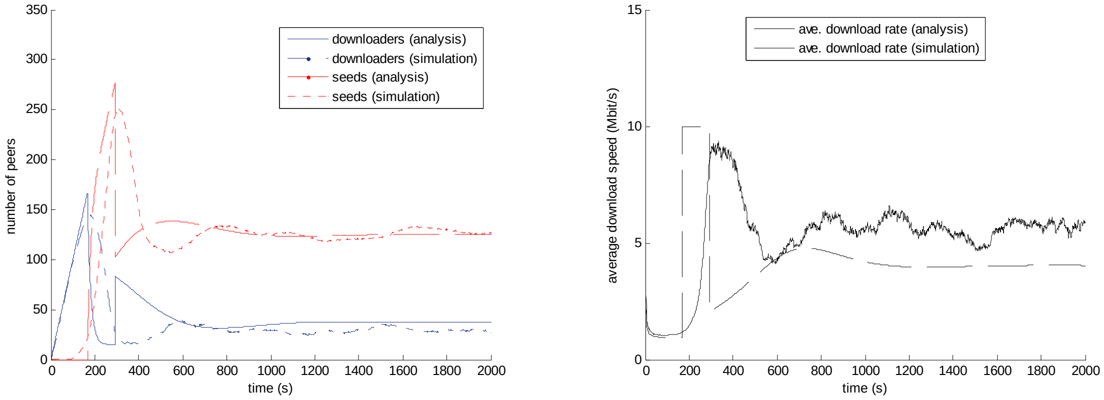


Figure 6.4: The number of downloaders $x_i(t)$ and seeds $y_i(t)$ (left) and the average download speed per peer $u_i(t)$ (right) as a function of time in a linearized system with γ_i=0.008.

In this case with $\gamma_i > \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os}}$, not only the number of seeds, but also the average download rate is smaller than the cases before, which leads to worse system performance.

We can observe from both analytical results and simulation results that the average download rate is not stable any more, but decreases sharply at around 400 seconds; even though it rebounds after that, it has a smaller value than in the previous cases. In Fig. 6.4, because the peers cannot download at full speed after 400 seconds in simulations, the peer correlation becomes critical for the downloaders to obtain the desired chunks. As a result, the differences between the fluid model and the discrete simulations increase. Nevertheless, the trends of the simulation results and the analytical results remain similar.

For both case studies, the lowest value of the avg. download rate appears in the Start-up phase (first 167 seconds), but it is still larger than the video playback rate so that the whole system performs well and no blocking will be experienced by end users.

6.6 Conclusion

In this Chapter, we have modeled mesh-based P2PVoD. This model, which is based on current P2PVoD applications, leads to non-linear system equations. In a non-linear model, small perturbations of the input may lead to large (undesirable) changes in the behavior of the system. Consequently, we have provided rules for a P2PVoD application that ascertain a linear behavior. A critical factor is the seed definite leaving rate γ_i. The best choice for P2PVoD application developers is to set it close to, but not exceeding, the value of $\frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os}} - \theta_i$.

With our model, parameters that affect the system performance were observed and analyzed. The results from realistic simulations match well with our analytical model. Our model can thus be used to predict the system behavior, which can aid in the design of P2PVoD systems.

Chapter 7

P2PVoD system with helpers

The idea of utilizing helpers' free upload capacity was first introduced by Wong [47]. Some idle users with spare upload capacity who are not interested in any particular file can be utilized as helpers to significantly improve the download speed beyond what can be achieved in a conventional BitTorrent network. Helpers represent a rich resource of untapped upload bandwidth which can be exploited for increasing the total system upload bandwidth and hence easing the bottleneck. The usage of helpers in BitTorrent-like file-sharing systems was discussed in [48]. Here, we discuss the usage of helpers in P2PVoD systems.

7.1 Limitations of P2PVoD system without helpers

In the previous Chapter, we did not find any problems in the system because we set parameters to $v < bw_{up}$. The reason behind this will be explained and analyzed in Appendix A.2. However, what will happen if $v > bw_{up}$?

For example, if too many ADSL peers are watching the video i (e.g., if we change $bw_{up} = 0.9$ Mbit/s to $bw_{up} = 0.4$ Mbit/s $< v$ in our case studies), or if a HDTV video is broadcasted (if we change $v = 0.5$ Mbit/s to 1 Mbit/s $> bw_{up}$ in our case studies), will the system still perform well? We found that the system will indeed face problems, which will be discussed in the following.

7.1.1 Start-up phase problem

Here, the peers' bad performance in the start-up phase when $v > bw_{up}$ is referred to as the start-up phase problem.

In the start-up phase of the system ($0 \leq t < \tau_i$, where $\tau_i \approx \frac{L_i v}{bw_{up}} > L_i$), peers cannot download the video data in time before the playback deadline and the whole system will experience bad performance.

The system behavior and performance when $v > bw_{up}$ can be further seen in the following case study.

We let $v = 1$ Mbit/s $> bw_{up}$ and $\gamma_i = 0.0033 < \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - (L_i v \lambda_i bw_{up}/bw_{down})}$, but keep other parameters the same as in Section 6.5.2. We can see from Fig. 7.1 that during the Start-up phase (first 340 seconds, much longer than that in Fig. 6.3), the average download speed at a peer $u_i(t)$ is smaller than the playback rate v, which means most peers face video blocking in this case. But afterwards, more and more seeds appear and stay in the system long enough ($\frac{1}{\gamma_i} \approx 303$ seconds). In this way, the resources of the system become larger so that in steady state the download bandwidth at peers can be fully used and the whole system recovers and performs well.

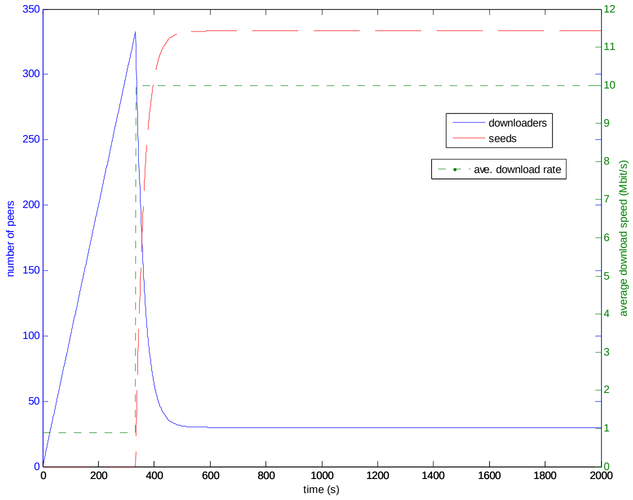


Figure 7.1: The number of downloaders $x_i(t)$ and seeds $y_i(t)$ and the average download speed per peer $u_i(t)$ as a function of time in a linearized system with $v = 1$ Mbit/s $> bw_{up}$ and γ_i=0.0033.

7.1.2 Burden on peers

When the network upload bandwidth resources are limited (e.g. $v > bw_{up}$), peers have to keep and share the video content stored at them for a long time even after finishing viewing the video. In this case, the burden on a peer is quite large. More importantly, it is hard to achieve in reality. If a peer decides to leave from the system after the video ends (frequently happens in reality), but its seed serving time is still smaller than $\frac{\lambda_i L_i v - bw_{os} - \frac{L_i v \lambda_i bw_{up}}{bw_{down}}}{\lambda_i bw_{up}}$ seconds, the software developer has no good way to continue making this peer share its content any more.

7.2 Solution: Adding helpers into the system

We propose a new system with helpers. We will explain how to define and set the helpers' behavior so that the start-up phase problem can be solved and at the same time the burden (hard disk and bandwidth usage) at each normal peer can be alleviated to some extent by adding helpers into the system.

In this system with helpers, each helper (e.g. a peer who is active in the system but not viewing the video, and possibly with larger upload capacity) only downloads a small number of useful chunks of the file and actively searches for peers who do not possess these chunks to upload to.

Before a helper finishes his downloading, he is a helper downloader. Once he finishes the download, he becomes a helper seed. We propose that a helper stops downloading after obtaining a small size of the file, C_h (Mbits).

A helper downloader, on one hand, can contribute his upload bandwidth to the whole system; but on the other hand, he also costs the system resources to download some video chunks from normal peers or other helpers or the original source provider. A helper seed does not download any more, but uploads his video content to both helper downloaders and normal peer downloaders.

A detailed strategy for helpers is further described in Appendix B.

The role and the behavior of normal peers in the system remains the same.

To analyze the system with helpers, besides the symbols listed in Table 6.1, we introduce additional symbols in Table 7.1.

Table 7.1: Symbols for helpers

λ_i^h: Helper downloaders' arrival rate for video i.
θ_i^h: Helper's random leaving rate from video i.
γ_i^h: Helper seed's definite leaving rate from video i.
$x_i^h(t)$: No. of helper downloaders in the video i system at t.
$y_i^h(t)$: No. of helper seeds in the video i system at t.
bw_{up}^h: Avg. upload rate at a helper for video delivery.
bw_{down}^h: Max download rate of a helper downloader for video delivery.
$u_i^h(t)$: Avg. download rate of a helper downloader at t in video i system.

In this system with helpers, we assume that when the normal peers' upload bandwidth is the constraint, the helpers' upload bandwidth is also the constraint. If the resources are not enough for normal peers, there are no extra resources for the helpers. On the other hand, in the case that the overall upload rate resources are enough, we can assume that when the helpers' download bandwidth can be fully used, normal peers' download bandwidth can also be fully used.

Hence, we take only two system situations into consideration: when the upload rate resources are enough (the download bandwidth is the constraint for both normal peers and helpers) or when the upload rate resources are limited (the upload bandwidth is the constraint for both normal peers and helpers). Our system equations transform as follows:

$$\begin{aligned}&u_i(t)x_i(t) + u_i^h(t)x_i^h(t) = \\ &\min\{bw_{down}x_i(t) + bw_{down}^h x_i^h(t), bw_{up}(x_i(t)+y_i(t))+bw_{os}+bw_{up}^h(x_i^h(t)+y_i^h(t))\}\end{aligned} \tag{7.1}$$

$$\frac{dx_i(t)}{dt} = \lambda_i - \theta_i x_i(t) - \frac{u_i(t)x_i(t)}{L_i v} \tag{7.2}$$

$$\frac{dy_i(t)}{dt} = \frac{u_i(t)x_i(t)}{L_i v} - \gamma_i y_i(t) \tag{7.3}$$

$$\frac{dx_i^h(t)}{dt} = \lambda_i^h - \theta_i^h x_i^h(t) - \frac{u_i^h(t)x_i^h(t)}{C_h} \tag{7.4}$$

$$\frac{dy_i^h(t)}{dt} = \frac{u_i^h(t)x_i^h(t)}{C_h} - \gamma_i^h y_i^h(t) \tag{7.5}$$

When the resources in the system are enough, $u_i(t)x_i(t)+u_i^h(t)x_i^h(t) = bw_{down}x_i(t)+bw_{down}^h x_i^h(t)$; while when the resources are limited, we have $u_i(t)x_i(t) + u_i^h(t)x_i^h(t) = bw_{up}(x_i(t)+y_i(t))+bw_{os}+bw_{up}^h(x_i^h(t)+y_i^h(t))$. If $u_i^h(t)$ is given in our model (can be set as a constant value smaller than bw_{down}^h in the start-up phase of the system, and equal to bw_{down}^h in the steady state of the system, which will be discussed later), we can solve the 5 system equations above to get out the 5 unknown parameters: $x_i(t)$, $y_i(t)$, $u_i(t)$, $x_i^h(t)$ and $y_i^h(t)$.

7.2.1 Solving the Start-up phase problem when $v > bw_{up}$

In order to let the average download speed at a peer be larger than his video playback rate ($u_i(t) > v$), we need to let

$$\begin{aligned}&\min\{bw_{down}x_i(t)+bw_{down}^h x_i^h(t), bw_{up}(x_i(t)+y_i(t))+ \\ &bw_{os} + bw_{up}^h(x_i^h(t) + y_i^h(t))\} - u_i^h(t)x_i^h(t) > vx_i(t)\end{aligned} \tag{7.6}$$

In the start-up phase ($0 \le t < \tau_i$) of the system, we already knew that $y_i(t) = 0$, $x_i(t) = \lambda_i t$ and $\tau_i = \frac{L_i v}{u_i(t)}$, as described in Section 6.3.2. The number of helper downloaders $x_i^h(t)$ and the number of helper seeds $y_i^h(t)$ can soon reach their steady state when C_h is very small[1], and we obtain the values of $\bar{x}_i^h$ and $\bar{y}_i^h$ in their steady state by setting Eqs. (7.4) and (7.5) equal to 0.

On the other hand, because in the start-up phase of the system there are no seeds purely helping the system, $bw_{down} > v > bw_{up}$ means that the peers' upload bandwidth is the constraint. Then, the helpers' upload bandwidth is also the constraint.

Then, Eq. (7.6) changes to

$$bw_{up}\lambda_i t + bw_{os} + bw_{up}^h \frac{u_i^h(t)\lambda_i^h + \lambda_i^h C_h \gamma_i^h}{\gamma_i^h u_i^h(t)} - \lambda_i^h C_h > v\lambda_i t$$

We already discussed before that, when the helpers' upload bandwidth is the constraint in the start-up phase of the system, we have $u_i^h(t) < bw_{down}^h$. Hence, the condition can be further expressed as

$$\lambda_i^h > \frac{(v - bw_{up})\lambda_i t - bw_{os}}{bw_{up}^h(\frac{1}{\gamma_i^h} + \frac{C_h}{bw_{down}^h}) - C_h}$$

Furthermore, because $t < \tau_i$, $\tau_i = \frac{L_i v}{u_i(t)}$ and we suppose $u_i(t) > v$, we have $t < L_i$ in the Start-up phase.

Thus, to make sure $u_i(t) > v$ in the start-up phase of the system when $v > bw_{up}$, we can set the helper arrival rate λ_i^h as a constant value which satisfies

$$\lambda_i^h > \frac{(v - bw_{up})\lambda_i L_i - bw_{os}}{bw_{up}^h(\frac{1}{\gamma_i^h} + \frac{C_h}{bw_{down}^h}) - C_h} \tag{7.7}$$

We can deduce from condition (7.7) that:

- If helper seeds stay in the system longer ($> \frac{1}{\gamma_i^h}$), less helpers are needed ($< \lambda_i^h$). Accordingly, a smaller value of $\frac{1}{\gamma_i^h}$ will lead to a larger value of $\lambda_i^h(t)$, which means more helpers will be needed[2].

  By setting and balancing the helper arrival rate λ_i^h and the time $\frac{1}{\gamma_i^h}$ helper seeds stay in the system, we can make sure $u_i(t) > v$ and solve the normal peers' bad performance problem in the start-up phase of the system.

---

[1] The time interval of the helpers' start-up phase is equal to $\frac{C_h}{u_i^h(t)}$. The analysis in [48] states that even with $C_h \ll L_i v$, the helpers' efficiency in distributing a file is almost as good as if they have the entire video file. Hence, C_h can be set $\ll L_i v$. Then, if the value of $u_i^h(t)$ is not too small ($u_i^h(t)$ is *not* $\ll u_i(t)$), we have $\frac{C_h}{u_i^h(t)} < \tau_i$ and the very small value of C_h determines the short length of the Start-up phase of helpers.

[2] This is reasonable because the shortage of network resources can be complemented by either adding more helpers, each of which contributes for a shorter time; or adding less helpers, each of which contributes for a longer time.

- If, we assume that $(v - bw_{up})\lambda_i L_i > bw_{os}$, which is easy to satisfy[3]. Then, in order to have a positive value of λ_i^h, we should satisfy:

  $$bw_{up}^h(\frac{1}{\gamma_i^h} + \frac{C_h}{bw_{down}^h}) > C_h \qquad \text{(condition A)}$$

  On the other hand, we can also consider this problem from another angle. If helpers are able to really help this system, overall they should upload more data than they download. Then we can get:

  $$bw_{up}^h(\frac{1}{\gamma_i^h} + \frac{C_h}{u_i^h(t)}) > C_h \qquad \text{(condition B)}$$

  We know that once condition A is satisfied, condition B is satisfied.

  Thus, the threshold of the helper seed serving time can be expressed as

  $$\frac{1}{\gamma_i^h} > \frac{(bw_{down}^h - bw_{up}^h)C_h}{bw_{down}^h bw_{up}^h} \tag{7.8}$$

  where $bw_{down}^h > bw_{up}^h$ agrees with reality.

- Because in reality users do not usually stay in the system as helpers for too long, $\frac{1}{\gamma_i^h}$ should be set as small as possible and at the same time satisfy the prerequisite of Eq. (7.8). Hence, we should make the C_h video content at a helper seed be shared for at least, but close to, $\frac{(bw_{down}^h - bw_{up}^h)C_h}{bw_{down}^h bw_{up}^h}$ seconds after his downloading of C_h finishes.

Then, parameter λ_i^h represents how many helpers are needed in the system, which can be set based on Eq. (7.7) and parameter $\frac{1}{\gamma_i^h}$ means how long a helper should continue to stay after his download ends, the value of which can be set based on Eq. (7.8).

Within the start-up phase of the system, $u_i(t)$ can be expressed as

$$u_i(t) = bw_{up} + \frac{bw_{os} + \lambda_i^h bw_{up}^h(\frac{1}{\gamma_i^h} + \frac{C_h}{u_i^h(t)}) - \lambda_i^h C_h}{\lambda_i t}$$

---

[3] Based on a measurement study (in the P2P streaming system PPLive), the peer arrival rate to a movie λ_i is at least $0.2/s$ [2, Fig. 11].

7.2.2 Usefulness of helpers in Steady state

We analyzed how to set the parameters for helpers to improve the performance of the start-up phase of the system in the previous section. In this section, we will discuss how to set the normal peers' parameters in steady state.

From the analysis in Section 6.3 we know that we can fully use the download bandwidth and let the system perform well and stable by setting $\gamma_i \leq \frac{\lambda_i bw_{up}}{\lambda_i L_i v - bw_{os} - \frac{L_i v(\lambda_i bw_{up} + \theta_i bw_{os})}{bw_{down}}}$. This condition can be reached by solving $\frac{L_i v}{bw_{down}} \geq \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$ (see the analysis of condition 1 in Section 6.3.5). Once $\frac{L_i v}{bw_{down}} \geq \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$, we will always have system equations (6.6) and (6.7) in steady state.

Like the deduction for the system without helpers, we can deduce our new condition for the system with helpers:

In steady state of the system, $y_i(t)$ is not equal to 0 and $x_i(t)$ is not equal to $\lambda_i t$ anymore. As discussed in Section 7.2.1, when the normal peer equations (7.2) and (7.3) reach the steady state, the helper equations (7.4) and (7.5) have already reached the steady state.

Condition 1 in Section 6.3.5 becomes $\frac{bw_{down}\bar{x}_i + bw^h_{down}\bar{x}^h_i}{L_i v} \leq \frac{bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os} + bw^h_{up}(\bar{x}^h_i + \bar{y}^h_i)}{L_i v}$. Based on a similar deduction process (see Appendix A.3), we can get the seed serving time $\frac{1}{\gamma_i}$ in the system with helpers as follows,

$$\frac{1}{\gamma_i} \geq \frac{Q}{\lambda_i bw_{up}} \tag{7.9}$$

where $Q = \lambda_i L_i v + \lambda^h_i C_h - \frac{\lambda^h_i C_h bw^h_{up}}{u^h_i(t)} - bw_{os} - \frac{bw^h_{up}\lambda^h_i}{\gamma^h_i} - \frac{bw_{up}}{bw_{down}}(\lambda_i L_i v - \frac{\lambda^h_i C_h bw^h_{down}}{u^h_i(t)} + \lambda^h_i C_h)$.

The download bandwidth can be fully used in steady state once the value of γ_i satisfies the condition (7.9).

We know that a larger value of γ_i will make the system reach the steady state faster, so our best choice is to set the value of $\frac{1}{\gamma_i}$ close to the right hand side of Eq. (7.9).

Comparable to Eq. (7.9) in the system *with* helpers, Eq. (6.8) gave the seed serving time threshold in the system *without* helpers. In steady state, in which system the value of $\frac{1}{\gamma_i}$ is larger, and how much larger/smaller depends on the difference

$$D = \frac{\frac{\lambda^h_i C_h bw^h_{up}}{u^h_i(t)} + \frac{bw^h_{up}\lambda^h_i}{\gamma^h_i} + \frac{bw_{up}}{bw_{down}}\lambda^h_i C_h(1 - \frac{bw^h_{down}}{u^h_i(t)}) - \lambda^h_i C_h}{\lambda_i bw_{up}}$$

We can input the value range of $\frac{1}{\gamma^h_i}$ in Section 7.2.1 into the expression of D and we find that this difference value is larger than $\frac{\lambda^h_i C_h bw^h_{up}(\frac{1}{u^h_i(t)} - \frac{1}{bw^h_{down}})}{\lambda_i bw_{up}} + \frac{\lambda^h_i C_h(1 - \frac{bw^h_{down}}{u^h_i(t)})}{\lambda_i bw_{down}}$, which is equal to 0 in steady state[4].

Hence, we can conclude that the seed serving time in the system *without* helpers minus the seed serving time in the system *with* helpers is larger than 0. As a result, in steady state when the download bandwidth can be fully used, each normal peer needs to keep and share the video D seconds less on average when we add helpers into our system.

Consequently, with adding helpers into the system, the value of $\frac{1}{\gamma_i}$ will be smaller so that the burden of each normal peer will be reduced in steady state. How much it can be reduced depends on the value of D.

7.2.3 Case study of the system with helpers when $v > bw_{up}$

We can feed the parameters into our mathematical model and solve the system Eqs. (7.1) to (7.5). Comparable with Table 6.2 in Section 6.5, here we only list parameters which are new or with changed values in Table 7.2.

Table 7.2: New value of parameters in this case study.

$v = 1Mbit/s$ (for a video with HDTV quality),
$bw_{up}^h = 1.1Mbit/s$,
$bw_{down}^h = 10Mbit/s$,
$C_h = 2Mbits$ (based on our helper policy we designed),
$\gamma_i^h = 0.5$ (which satisfies condition (7.8)),
$\lambda_i^h = 62$ (which satisfies condition (7.7)),
$u_i^h(t) = 1Mbit/s$ in Start-up phase of the system;
$u_i^h(t) = 10Mbit/s$ in Steady state of the system,
$\gamma_i = 0$ in Start-up phase;
$\gamma_i = 0.0037$ in Steady state (which satisfies condition (7.9))

In Table 7.2, the values of λ_i^h and γ_i^h represent that much more helpers than the real video viewers appear in this system within a certain time interval, but these helpers only need to stay in the system to purely contribute for around $\frac{1}{\gamma_i^h} = 2$ seconds. Besides, $\gamma_i = 0.0037$ represents that a normal peer needs to serve the system as a seed for $\frac{1}{\gamma_i} \approx 270$ seconds. By adding helpers into the whole system, an average of 33 seconds resource burden at each normal peer can be reduced in this case, compared to the $\frac{1}{\gamma_i} \approx 303$ seconds in the system without helpers (see Section 7.1). Nevertheless, there is not much burden added to the helpers (only an extra 2 seconds burden will be added to each helper in this case).

---

[4] This is because, once we can fully use the download bandwidth, Eq. (7.1) becomes $u_i(t)x_i(t) + u_i^h(t)x_i^h(t) = bw_{down}x_i(t) + bw_{down}^h x_i^h(t)$, where $u_i^h(t) = bw_{down}^h$ can be set in steady state and we also have $u_i(t) = bw_{down}$ accordingly.

As discussed before, we can know from Section 7.2.1 and 7.2.2 that: 1) In the Start-up phase of the whole system, the resources are limited and we use $u_i(t)x_i(t) + u_i^h(t)x_i^h(t) = bw_{up}(x_i(t)+y_i(t))+bw_{os}+bw_{up}^h(x_i^h(t)+y_i^h(t))$, where $u_i^h(t)$ can be assumed as a constant value; 2) In steady state of the whole system, the resources are enough and we use $u_i(t)x_i(t)+u_i^h(t)x_i^h(t) = bw_{down}x_i(t)+bw_{down}^h x_i^h(t)$, where $u_i^h(t)$ can be equal to bw_{down}^h.

The situation of normal peers in the system with helpers in this case is shown in Fig. 7.2.

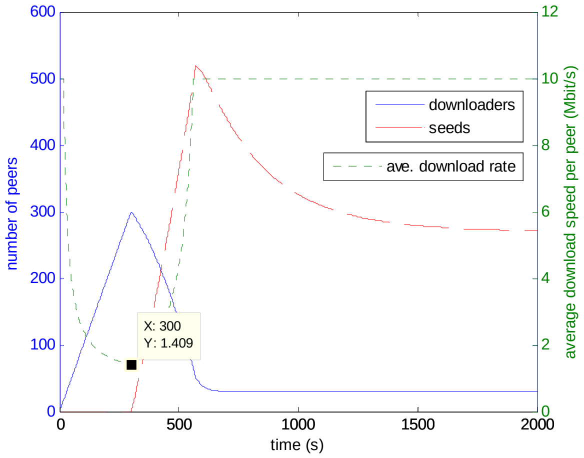


Figure 7.2: The number of downloaders $x_i(t)$ and seeds $y_i(t)$ and the average download speed per peer $u_i(t)$ as a function of time in the linearized system with helpers when $v = 1$ Mbit/s $> bw_{up}$ and γ_i=0.0037.

In this case study, besides the results shown in Fig. 7.2, we also obtained the results regarding helpers: the number of helper downloaders and the number of helper seeds in the Start-up phase both quickly (around 2 seconds) reach to 124; while in the steady state, the number of helper downloaders is 13 and the number of helper seeds is 124.

We can observe from Fig. 7.2 that during the whole period, the average download speed per peer $u_i(t)$ is at least 1.409 Mbit/s, which means that we always have $u_i(t) > v$ as long as the rules of setting parameters are followed.

Compared to the system with the same settings for our parameters, but without helpers (shown in Fig. 7.1 in Section 7.1, where $u_i(t) < v$ in Start-up phase), adding helpers into the system can indeed solve the Start-up phase problem of the system when $v > bw_{up}$. On the other hand, in the system with helpers, the burden on peers can be much reduced without adding much burden to the helpers.

7.3 Conclusions

The model analyzed in Chapter 6 had some limitations. That is, we have to set the video playback rate v smaller than the average upload rate at a user bw_{up} in order to have a good system performance *all the time*. If $v > bw_{up}$, we will not only have a so-called start-up phase problem but also there will be too much burden on peers. To solve these problems, we proposed an improved system with helpers. With our analysis of this new system, we deduced the rules of setting some key parameters (e.g. the new value of the seed serving time, the helper seed serving time and the helper arrival rate) to make sure that the average download rate of a user is always larger than the fixed video playback rate. Moreover, in order to optimize the upload bandwidth resource usage, strategies for helpers were also proposed.

Once our novel designs are employed and the conditions of our parameters are satisfied, we can have an optimal and stable system performance all the time under all conditions, while optimizing the system resource usage as well.

Part III

Video Conferencing (VC) Systems

Chapter 8

VC Background

8.1 Classification of VC

VC can be classified according to its different terminals, user interfaces, signaling protocols, and network structures.

8.1.1 Different terminals

1. Desktop VC: Add-ons to normal PCs with Internet access, transforming them into VC devices. Convenient and low-cost, but quality cannot be guaranteed. In this thesis, we will focus on this kind of VC systems.

2. Dedicated VC System: All required components are packaged into a single piece of equipment or in a room. It has high-quality video and audio, but it is fixed and very expensive. In TU Delft, there is a Room VC system located at Mekelweg 2. It uses a POLYCOM HDX9000 videoconference system, which costs 40000 USD, and 125 euro/hour to use. It can support six participants simultaneously.

3. IP VideoPhone: Only point-to-point communication.

8.1.2 User interface

The user interface is the connection between the user and the VC systems. It enables the user to log in, schedule new meetings, attend meetings, and get some in-conference controls. In general, there are two kinds of user interface:

1. Web browser interface: Users can schedule a new meeting, or join an existing meeting through a web browser without installing any software.

2. Software user interface: The interface generally consists of a series of menus, allowing the caller to interact with the system based on a set of context-sensitive scripts.

8.1.3 Signaling protocol

1. H.323 is an umbrella Recommendation from the ITU Telecommunication Standardization Sector (ITU-T) that defines the protocols to provide audio-visual communication sessions on any packet network. The H.323 standard addresses call signaling and control, multimedia transport and control, and bandwidth control for point-to-point and multi-point conferences. H.323 was the first VoIP standard to adopt the IETF standard RTP to transport audio and video over IP networks. H.323 defines several network elements including endpoint, gateways to allow interworking between IP network and other network types (e.g. PSTN) because H.323 specification also includes the H.245 protocol for the control channel used in circuit-switched networks like ISDN/PSTN and POTS.

2. The Session Initiation Protocol (SIP) is a TCP/IP-based Application Layer signaling protocol. It was designed by Henning Schulzrinne and Mark Handley in 1996. The latest version of the specification is RFC 3261 from the IETF Network Working Group. It is also possible to use the H.245 capability description elements in SIP to achieve interworking between packet-switch networks and circuit-switch networks.

   Hence, in general, H.245 in H.323 defines procedures that allow receivers to control media encoding, transmission rates, and error recovery. SIP, instead, relies on RTCP for providing feedback on reception quality. The feedback it provides automatically scales from a two person point-to-point conference to huge broadcast style conferences with millions of participants. Other comparisons between H.323 and SIP can be found in [49]. In [50], a video conferencing gateway supporting interoperability between SIP and H.323 was developed.

8.1.4 Network structure

1. In a centralized or Client-Server model, all the components of a conferencing system are implemented in a single server (or some servers). Each endpoint only communicates with the servers.

2. A distributed network can be further divided into a decentralized P2P network and a hybrid (centralized and decentralized) network. In distributed architectures, each service provides a logical functionality distributed among multiple physical

devices. There is a server or many servers for signaling, while the media data is delivered directly among endpoints.

8.2 Available Desktop VC applications

In order to choose suitable VC applications for further study, we made a survey of the available Desktop VC applications on the Internet.

We have considered eighteen VC applications, for which we list the maximum frame rate they can support (the best video quality they can provide), the maximum number of simultaneous conference participants, and the category (S/C or P2P) they belong to, in Table 8.1.

Table 8.1: Popular video conferencing applications.

	Max. frame rate (frames/second)	Max. # of simultaneous video participants	S/C or P2P
Eedo WebClass		6	web-based S/C
IOMeeting	30	10	web-based S/C
EarthLink	30	24	S/C
VideoLive	30	6	web-based S/C
Himeeting	17	20	S/C
VidSoft	30	10	S/C
MegaMeeting	30	16	web-based S/C
Smartmeeting	15	4	S/C
Webconference	15	10	web-based S/C
Mebeam		16	web-based S/C
Confest	30	15	S/C
CloudMeeting	30	6	S/C
Linktivity Web	30	6	web-based S/C
WebEx	30	6	web-based S/C
Nefsis Free Trial	30	10	S/C
Lava-Lava	15	5	decentralized P2P
Qnext		4	centralized P2P
Vsee	30	8	decentralized P2P

Besides the applications listed in Table 8.1, *Vmukti* is a Multi-party video and audio media call center software, which uses a P2P tree-based methodology to distribute content. It is an open source application[1].

---

[1] *Vmukti* can be downloaded from http://sourceforge.net/projects/vmukti/files/

Even though there exist many free VC applications, many of them turn out to be instable once installed. From Table 8.1, we observe that the maximum frame rate is 30 frames/s, which corresponds to regular TV quality. All applications support only a very limited number of participants and the applications that support more than 10 simultaneous participants all use a centralized S/C network structure.

Many other popular online chatting applications (like Skype, MSN, Yahoo messenger, Google talk, etc.) only support multi-party audio conference and 2-party video conference, and therefore are not considered here.

Chapter 9

Measurement Study of Multi-Party Video Conferencing

In this Chapter, we take four representative video conferencing applications and reveal their characteristics and different aspects of Quality of Experience. Based on our observations and analysis, we recommend to incorporate the following aspects when designing video conferencing applications: 1) Traffic load control/balancing algorithms to better use the limited bandwidth resources and to have a stable conversation; 2) Use traffic shaping or adaptively re-encode streams in real time to limit the overall traffic.

This work as published in [51] is, to our knowledge, the first measurement work to study and compare mechanisms and performance of existing free multi-party video conferencing systems.

9.1 Introduction

In this Chapter, we focus on studying *free* applications that provide multi-party (≥ 3 users) VC on the Internet, and focus on the following questions:

- *How do multi-party VC applications work?*
- *How much resources do they need?*
- *What is the Quality of Experience (QoE)?*
- *What is the bottleneck in providing multi-party VC over the Internet?*
- *Which technology and architecture offer the best QoE?*

In order to answer these questions we chose and measured four representative VC applications to investigate, namely *Mebeam*[1], *Qnext*[2], *Vsee*[3], and *Nefsis*[4].

9.2 Related work

Most research focuses on designing the network architectures, mechanisms and streaming technologies for VC. In this section we only discuss the work on studying and comparing the mechanisms and performance of streaming applications.

Skype supports multi-party audio conferencing and 2-party video chat. Baset and Schulzrinne [52] analyzed key Skype functions such as login, call establishment, media transfer and audio conferencing and showed that Skype uses a centralized P2P network to support audio conferencing service. Cicco *et al.* [53] measured Skype video responsiveness to bandwidth variations. Their results indicated that Skype video calls require a minimum of 40 kbps available bandwidth to start and are able to use as much as 450 kbps. A video flow is made elastic through congestion control and an adaptive codec within that bandwidth interval.

Microsoft Office Live Meeting (Professional User License) uses a S/C architecture and has the ability to schedule and manage meetings with up to 1,250 participants. However, only few participants can be presenters who can upload their videos and the others are non-active attendees.

Spiers and Ventura [54] implemented IP multimedia subsystem (IMS)-based VC systems with two different architectures, S/C and P2P, and measured their signaling and data traffic overhead. The results show that S/C offers better network control together with a reduction in signaling and media overhead, whereas P2P allows flexibility, but at the expense of higher overhead.

Silver [55] discussed that applications built on top of web browsers dominate the world of Internet applications today, but are fundamentally flawed. The problems listed include delays and discontinuities, confusion and errors, clumsy interfacing and limited functionality.

Trueb and Lammers [56] analyzed the codec performance and security in VC. They tested High Definition (HD) VC and Standard Definition (SD) VC traffic characteristics and their corresponding video quality. In their results, HD provides a better video quality at good and acceptable network conditions, while in poor network conditions HD and SD have similar performance.

As we know, no articles compared the different types of existing free multi-party VC systems or measure their QoE. In this Chapter, our aim is to provide such a comparison.

---

[1] http://www.mebeam.com/

[2] http://www.qnext.com/

[3] http://www.vsee.com/

[4] http://www.nefsis.com/leads/free-trial.aspx

9.3 Experiments Set-up

We have chosen four representative applications to study and use their default settings:

- *Mebeam*: web-browser based S/C with a single server center.
- *Qnext (version 4.0.0.46)*: centralized P2P. The node which hosts the meeting is the super node.
- *Vsee (version 9.0.0.612)*: decentralized full-mesh P2P.
- *Nefsis (free trial version)*: S/C, network of distributed computers as servers.

We have chosen these four applications because they each represent one of the four architectures under which all eighteen applications in Table 8.1 can be classified.

We have performed two types of experiments: (1) local lab experiments, composed of standard personal computers participating in a local video conference, in order to investigate the login and call establishment process, as well as the protocol and packet distribution of the four VC applications; (2) global experiments, to learn more about the network topology, traffic load and QoE, when a more realistic international video conference is carried out.

The global measurements were conducted during weekdays of May, 2009, under similar and stable conditions[5]:

- Client 1: 145.94.40.113; TUDelft, the Netherlands; 10/100 FastEthernet;
  Client 2: 131.180.41.29; Delft, the Netherlands; 10/100 FastEthernet;
  Client 3: 159.226.43.49; Beijing, China; 10/100 FastEthernet;
  Client 4: 124.228.71.177; Hengyang, China; ADSL 1Mbit/s.
- Client 1 always launches the video conference (as the host);
- Clients 1, 3 and 4 are behind a NAT.

To retrieve results, we used the following applications at each participant:

- *Jperf* to monitor the end-to-end available bandwidth during the whole process of each experiment. We observed that usually the network is quite stable and that the available end-to-end bandwidth is large enough for different applications and different participants.

---

[5] We have repeated the measurements in July, 2009 and obtained similar results to those obtained in May 2009.

- *e2eSoftVcam* to stream a stored video via a virtual camera at each VC participant. Each virtual camera is broadcasting in a loop a "News" video (.avi file) with a bit rate of 910 Kbit/s, frame rate of 25 frames/s and size 480x270;
- *Camtasia Studio 6*. Because all applications use an embedded media player to display the Webcamera streaming content, we have to use a screen recorder to capture the streaming content. The best software available to us was *Camtasia*, which could only capture at 10 frames/s. In order to have a fair comparison of the original video to the received video, we captured not only the streaming videos from all participants, but also the original streaming video from the local virtual camera[6].
- *Wireshark* to collect the total traffic at each participant.

9.4 Measurement results

9.4.1 Login and Call establishment process

Mebeam: We open the *Mebeam* official website to build a web video-chat room and all participants enter the room. The traces collected with *Wireshark* revealed that two computers located in the US with IP addresses 66.63.191.202 (Login Server) and 66.63.191.211 (Conference Server) are the servers of *Mebeam*. Each client first sends a request to the login server, and after getting a response sets up a connection with the single conferencing server center. When the conference host leaves from the conference room, the meeting can still continue. *Mebeam* uses TCP to transfer the signals, and RTMP[7] to transfer video and audio data.

Qnext: The data captured by *Wireshark* reveals two login servers located in the US. Each client first sends packets to the login servers to join the network. After getting a response, they use SSLv3 to set up a connection with the login servers. In the call establishment process, each client communicates encrypted handshake messages with 3 signaling servers located in the US and Romania and then uses SSLv3 to set up a connection between the client and the signaling server. When client A invites another client B to have a video conference and client B accepts A's request, they use UDP to transfer media data between each other. In a conference, there is only one host and

---

[6] We assess the video quality using the full reference batch video quality metric (bVQM) which computes the quality difference of two videos. Capturing at 10 frames/s a video with frame rate of 25 frames/s may lead to a different averaged bVQM score. However, because the video used has a stable content (there are only small changes in the person profile and background), we do not expect a large deviation in bVQM score with that of the 25 frames/s video. The results are accurate for 10 frames/s videos.

[7] Real-Time Messaging Protocol (RTMP) is a protocol for streaming audio, video and data over the Internet, between a Flash player and a server.

other clients can only communicate with the host. The host is the super node in the network. When the host leaves the meeting, the meeting will end. If another node leaves, the meeting will not be affected. *Qnext* uses TCP for signaling and UDP for video communication among participants.

Vsee: Each client uses UDP and TCP to communicate with the web servers during the login process. In the call establishment process, after receiving the invitation of the host, each client uses[8] T.38 to communicate with each other. *Vsee* has many web servers: during our experiment, one in the Netherlands, one in Canada, and 7 located in the US. *Vsee* has a full-meshed P2P topology for video delivery. However, only the host can invite other clients to participant in the conference. When the host leaves the meeting, the meeting cannot continue. Other peers can leave without disrupting the meeting. *Vsee* is a video conferencing and real-time collaboration service. The communication among users is usually of the P2P type using UDP, with automatic tunneling through a relay if a direct connection is not available.

Nefsis: In the login process, the clients first use TCP and HTTP to connect to the Virtual Conference Servers (with IP addresses 128.121.149.212 in the US and 118.100.76.89 in Malaysia) and receive information about 5 other access points from the Virtual Conference Servers. These 5 access points are also the data server centers owned by *Nefsis*, and they are located in the Netherlands (Rotterdam and Amsterdam), in the UK, India, Australia, and Singapore. Afterwards, the clients choose some access points to set up connections via TCP. After entering the conference room, each client communicates with each other through the access point when firewalls/NAT are present at clients, otherwise clients can set-up an end-to-end connection to communicate with each other directly. *Nefsis* uses TCP for signaling and delivering streaming data.

9.4.2 Packet size distribution and traffic load

To differentiate between non-data, video and audio packets, we performed three local experiments for each application. The first experiment uses two computers with cameras and microphones to have a video conference. In the second experiment, two computers are only equipped with microphones, but without cameras (no video packets will be received). In the third experiment, two computers set-up a connection, both without microphones and cameras (so only non-data packets will be exchanged).

Based on *Wireshark* traces, we could distill for each VC application the packet size range as shown in Table 9.1:

Other interesting observations are: 1) If the person profile or background images in the camera change/move acutely, a traffic peak is observed in our traces. 2) The traffic does not necessarily increase as more users join the conference. Fig. 9.1 shows the change of the average traffic load at each user when a new participant joins the

---

[8] T.38 is an ITU recommendation for fax transmission over IP networks in real-time.

Table 9.1: The packet size distribution of *Mebeam*, *Qnext*, *Vsee* and *Nefsis*.

Packet size	*Mebeam*	*Qnext*	*Vsee*	*Nefsis*
Audio	> 50 bytes	72 bytes	100 ∼ 200 bytes	100 ∼ 200 bytes
Video	> 200 bytes	50 ∼ 1100 bytes	500 ∼ 600 bytes	1000 ∼ 1600 bytes
Signaling	50 ∼ 200 bytes	50 ∼ 400 bytes	50 ∼ 100 bytes	50 ∼ 100 bytes

conference[9]. The decreasing slope after 3 users indicates that *Mebeam*, *Qnext* and *Vsee* either re-encoded the videos or used traffic shaping in order to reduce/control the overall traffic load in the system. We can see from Fig. 9.1 that only the traffic load at *Nefsis* clients does not decrease when the number of video conferencing participants reaches to 4. Therefore, we introduced more participants into the video conference for *Nefsis*, and we found that the traffic at each *Nefsis* user starts to decrease at 5 participants. Hence, we believe that in order to support more simultaneous conference participants, the overall traffic has to be controlled.

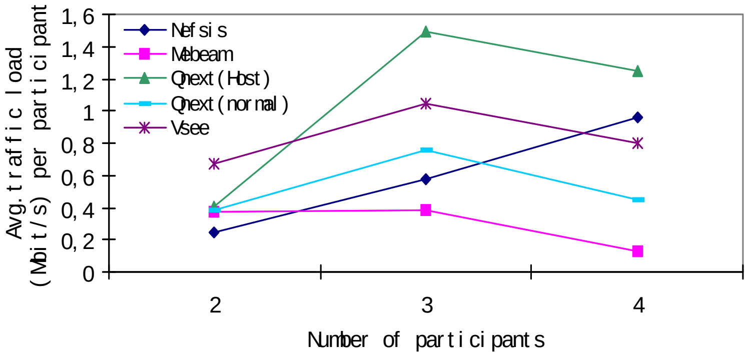


Figure 9.1: The average traffic load at an end-user when the number of conference participants increases from 2 to 4 (*Qnext* is limited to 4 participants).

Fig. 9.1 illustrates that, compared with the traffic generated by *Nefsis* which uses the same coding technology and the same frame rate on the same video, *Qnext* and *Vsee* generate most traffic, especially the host client of *Qnext*. This is because *Qnext* and *Vsee* use P2P architectures where the signaling traffic overhead is much more than the traffic generated by a S/C network with the same number of participants. The host client (super node) of *Qnext* generates 3 times more traffic than other normal clients. Hence, for this architecture, a super-node selection policy is recommended to choose a suitable peer (with more resources, for example) as the super node.

---

[9] We captured the packets after the meeting was set up and became stable.

Fig. 9.1 also shows that *Mebeam* generates the least traffic. Considering that the overall traffic load, which can be supported in a VC network, has an upperbound due to the limited users' bandwidth, and each *Mebeam* client generates much less traffic than the three other applications, it clarifies why *Mebeam* can support 16 simultaneous video users while *Nefsis* can only support 10 users, *Vsee* can only support 8 users and *Qnext* can only support 4 users.

9.4.3 Quality of Experience (QoE)

In this section, we assess via global measurements the QoE at the end user with respect to their video quality, audio-video synchronization, and level of interaction.

Video Quality

Similar to the approach used for assessing the image quality of P2PTV in Section 2.4.1, we use bVQM (Batch Video Quality Metric) to analyze the VC's video quality off-line in our objective measurements. bVQM takes the original video and the received video and produces quality scores that reflect the predicted fidelity of the impaired video with reference to its undistorted counterpart. The sampled video needs to be calibrated. The calibration consists of estimating and correcting the spatial and temporal shift of the processed video sequence with respect to the original video sequence. The final score is computed using a linear combination of parameters that describe perceptual changes in video quality by comparing features extracted from the processed video with those extracted from the original video. The bVQM score scales from 0 to approximately[10] 1. The smaller the score, the better the video quality.

We captured at every participant the stream from the imbedded multimedia player of each VC application with *Camtasia Studio 6*, and used *VirtualDub* to cut and synchronize the frames of the compared videos.

Table 9.2 provides the bVQM scores per participant for each VC service.

Table 9.2 indicates that *Nefsis* features the best video quality among the 4 applications, although with an average bVQM score of 0.61 (its quality is only "fair", which will be explained later with the subjective measurements). The highest bVQM score (the worst video quality) appears at Client 1 (the super node) of *Qnext*. Generally speaking, all four VC applications do not provide good quality[11] in this case.

---

[10] According to [22], bVQM scores may occasionally exceed 1 for video scenes that are extremely distorted.

[11] We also objectively measured the audio quality using metric PESQ-LQ (Perceptual Evaluation of Speech Quality-Listening Quality) [57] [21] and found that the PESQ-LQ average score (scale from 1.0 to 4.5, where 4.5 represents an excellent audio quality) is 2.24, 2.68, 3.08 and 3.15 for *Mebeam*, *Qnext*, *Vsee,* and *Nefsis,* respectively.

Table 9.2: The video quality of *Mebeam*, *Qnext*, *Vsee* and *Nefsis* at 4 clients.

VQM score	Client 1	Client 2	Client 3	Client 4	Average
Mebeam (Flash video, MPEG-4)	0.63	0.41	0.94	0.86	0.71
Qnext (MPEG-4, H.263, H.261)	1.05	0.94	0.63	0.83	0.86
Vsee	0.78	0.82	0.80	0.79	0.80
Nefsis (MPEG-4, H.263, H.263+)	0.34	0.61	0.61	0.87	0.61

Because no standard has been set for what level of bVQM score corresponds to what level of perceived quality of a VC service, we have also conducted subjective measurements. We use the average Mean Opinion Score (MOS) [21], a measure for user perceived quality, defined on a five-point scale[12]: 5 = *excellent*, 4 = *good*, 3 = *fair*, 2 = *poor*, 1 = *bad*.

We gave 7 different quality videos generated by VC applications to 24 persons who gave a subjective MOS score independently. We also objectively computed their bVQM scores. Fig. 9.2 shows the correlation between the objective bVQM scores and the subjective MOS values.

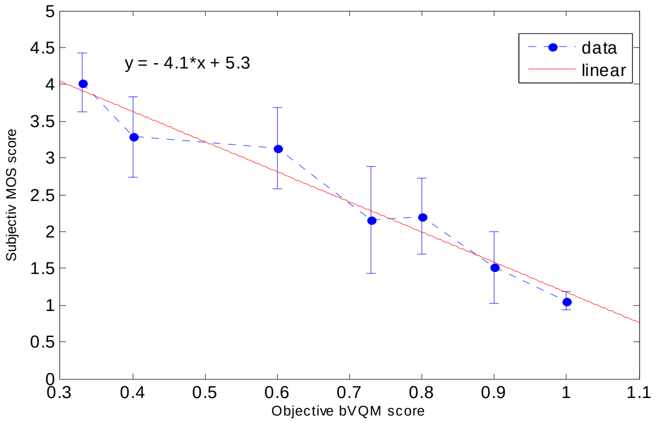


Figure 9.2: Relation between bVQM and MOS for video conferencing service.

We mapped between the bVQM scores and the average MOS scores over 24 persons, and found that they have a linear correlation in the range 0.3 <bVQM score≤ 1. Hence, the VC's video quality is predictable when using the objective metric bVQM.

---

[12] The threshold for acceptable TV quality corresponds to the MOS value 3.5.

Compared with the video quality of a global P2PTV distribution service, which has an average MOS value of 4 [58], the video quality of a global VC service is poor (with an average bVQM score of 0.74 and MOS value of around 2.2), because the VC service requires end users to encode and upload their streams in real-time. Even the local uploaded video has a largely degraded quality although it is still the best among all participants.

Audio-Video Synchronization

The relative timing of sound and image portions of a streaming content may not be synchronized. ITU [23] [59] has suggested that the viewer detection thresholds of audio-video lag are about +45 ms to −125 ms, and the acceptance thresholds are about +90 ms to −185 ms, for video broadcasting.

Similar to the approach used for assessing the audio-video synchronization of P2PTV in Section 2.4.1, we used an "artificially generated" video test sample, in which the video and audio waveforms are temporally synchronized with markers, to analyze the A/V synchronization provided by each VC application. We captured at each end user the videos from all other participants. When the audio and video tracks were extracted and compared off-line, there was an average difference in time between the two tracks of about 650 ms for *Mebeam*, 470 ms for *Qnext*, 400 ms for *Vsee* and 350 ms for *Nefsis*. Such large audio-video lags are mainly caused by a large amount of frame losses, which lead to the low video quality mentioned already in Section 9.4.3.

Interactivity (communication delay)

During a video conference it is annoying to have large communication delay[13]. Large communication delay implies lack of real-time interactivity in our global multi-party VC experiments. We measured the video delays among participants by injecting in the network another artificial video that mainly reproduced a timer with millisecond granularity.

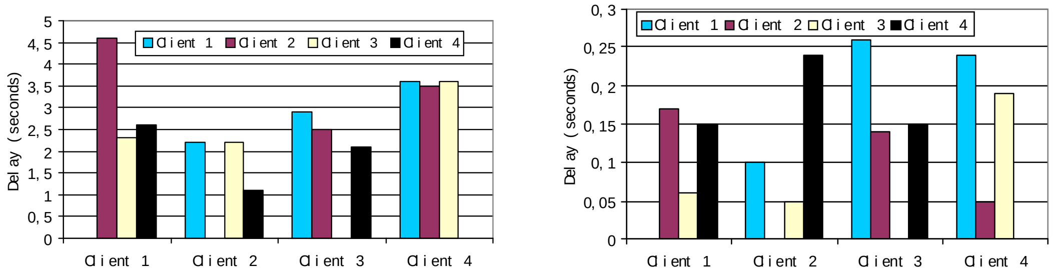
In the video conference, this artificial "timer" video was uploaded via the virtual camera and transmitted among the participants via the different VC applications. At each participant, we used a standard universal Internet time as reference[14]. We displayed the "timer" videos of all participants in real time. After a 1-minute long stable video conference, we cut the captured content at each participant with *VirtualDub* to compare the "timers" between any 2 participants. For each application, we took samples at 2 different times to calculate an average delay.

---

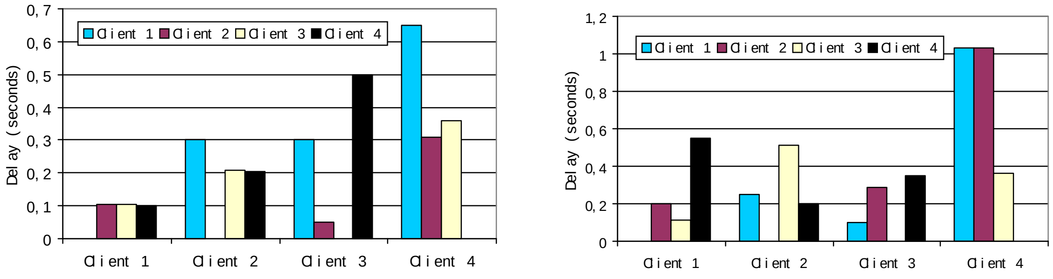
[13] In IP video conferencing scenarios, the maximum communication delay recommended by ITU is 400 ms [60].

[14] http://www.time.gov/timezone.cgi?Eastern/d/-5/java

The video delays among participants are shown in Fig. 9.3. The x axis shows the 4 different clients. The y axis shows the video transmission delay from the participant on the x axis to the participant shown in the legend.



(a) *Mebeam* (b) *Qnext*



(c) *Vsee* (d) *Nefsis*

Figure 9.3: The video delay between different participants.

Fig. 9.3 shows that *Qnext* provides a video that is most synchronized among the clients. *Qnext*, *Vsee*, and *Nefsis* have a comparable level of average video delay, respectively 0.15 s, 0.27 s, and 0.41 s. However, *Mebeam* clients suffer a huge video delay (2.77 s on average), because the processing time at the server is too long.

We also measured the audio delays among participants by injecting in the video an artificial DTMF (Dual-tone multi-frequency) tone. We sent and recorded the audio at Client 1. Other participants kept their speaker and microphone on, but did not produce extra audio. Based on the recorded audio tracks, we compared the time the audio marker was sent from Client 1 and the time the same audio marker was heard again at Client 1 after the transmitted audio was played, recorded, and retransmitted by a client. The time difference is approximately twice the one-way audio delay plus the processing delay at a client. Our results revealed 1 s, 1.4 s, 0.2 s and almost 0 s on average for *Mebeam*, *Qnext*, *Vsee* and *Nefsis* respectively. *Nefsis* performs best in the aspect of interactivity, while *Qnext* in this case provided the least synchronized audio among the users. When we measure the audio delay and the A/V synchronization, the delay is the end-to-end delay including the transmission delay and the delay introduced by the application. In our experiment, the video delay represents the delay of a same video scene that was captured at the application interfaces of the sender and the receiver, which does not include the time used for uploading the video to the sender via applications. Hence, considering the audio delay, video delay, and the A/V synchronization, we can conclude that the delay introduced by the application, when uploading, is large for *Qnext*.

9.4.4 Worst-case study

In another set of global experiments in June, 2009, our *Jperf* plots indicated that the end-to-end connections of clients 3 and 4 with the host were very unstable. We found that the two participants in China always passively disconnected from the conference or could not even log into *Mebeam*, *Nefsis* and *Qnext*. *Vsee* could still work, but the quality was awful, with bVQM scores close to 1.

In order to investigate the minimum bandwidth to support a video conference, we repeated many experiments adjusting the upload rate upper-bound (using[15] *Netlimiter*) at each participant for a particular VC application to test the user's upload bandwidth minimally required to launch a video conference.

For *Qnext*, the threshold is 50 Kbit/s. If an end user's available upload bandwidth is $<$ 50 Kbit/s, (s)he cannot launch *Qnext*. For *Vsee*, the threshold is 50 Kbit/s; for *Nefsis* it is 28 Kbit/s; and for *Mebeam* it is 5 Kbit/s, which we believe are the minimally supported streaming bit rates set by the applications.

9.5 Summary and Conclusions

Through a series of local and global experiments with four representative video conferencing systems, we examined their behavior and mechanisms, and investigated their login process, the call establishment process, the packet size distribution, transfer protocols, traffic load, delivery topology, and different aspects of Quality of Experience.

Our main conclusions from the measurement results on the traffic characteristics of four different video conferencing systems are: (1) The QoE of multi-party video conferencing is very sensitive to bandwidth fluctuations, especially in the uplink. Hence, an adaptive resolution/frame rate policy should be deployed; (2) When the number of participants increases, the traffic load at each participant does not always increase correspondingly (see Fig. 9.1), suggesting that re-encoding at the video or a traffic shaping policy takes place to control the overall traffic in the system.

Our QoE measurement results are summarized as: (1) Compared with the non interactive multimedia services (i.e. P2PTV [58]), existing Internet video conferencing

---

[15] *NetLimiter* is an Internet traffic control and monitoring tool designed for setting download/upload transfer rate limits for applications.

applications in general cannot provide very good quality to their end users (poor video and audio quality, large audio-video lag, and long communication delay in some cases); (2) Only a limited number of multimedia participants are supported and rarely high definition webcamera streaming is supported due to the limited available bandwidth or the limited processing capability; (3) The existing systems are not reliable in the worst cases. When the network is unstable or the available upload bandwidth is very limited (thresholds have been found), none of the applications work properly.

It seems that the Server-to-Client architecture with many servers located all over the world is currently the best architecture for providing video conferencing via the Internet, because it introduces the least congestion at both servers and clients. Load balancing and load control algorithms help the overall performance of the system and the codec used is important for the quality that end users perceive. The bottleneck to support video conferencing with more participants and high definition streams is the overhead traffic generated by them. To support more simultaneous participants in a single conferencing session, the traffic load has to be controlled/limited by using traffic shaping or re-encoding the video streams.

We have chosen four representative video conferencing systems for our study, but the measurement methodologies mentioned in this Chapter can also be applied to other video conferencing applications, which could be compared with our study in the future.

Part IV

Conclusions

Chapter 10

Research context, methods, and contributions

In this thesis, multimedia streaming services like IPTV (IP Television streaming), P2PTV (Peer-to-Peer Television streaming), CDN (Content Delivery Networking), P2PVoD (Peer-to-Peer Video on Demand), IPVC (IP Video Conferencing) and P2PVC (Peer-to-Peer Video Conferencing) are analyzed based on E2E Blocking, Quality of Experience of users, Network characteristics and Traffic usage, System Stability.

10.1 IPTV

For IPTV, an analytical model was set up. Our model is based on the configuration of an existing IPTV system (KPN's Mine TV in the Netherlands), where IP-layer multicast was used to deliver the TV streaming content. One of the most important measures of Quality of Experience, the end-to-end request blocking probability, was computed in our model. Two main causes of blocking were considered: 1) limited processing capability of a DSLAM, and 2) Insufficient available capacity from the DSLAM to the edge router. In the case study, we focused on a Dutch IPTV network for which we had obtained the parameter values (i.e., the TV channel popularity distribution, the link capacity, the number of TV channels, the number of subscribers, the percentage of active users in rush hour, the Poissonian arrival and departure process of users) from real data.

The results indicated that the less popular channels (higher channel index) have a considerably higher probability to be blocked than the more popular channels. In addition, we observed that the users' requests are less likely blocked if there are less available TV channels, if there are less users, or if users leave more frequently. Hence, based on our model, the probability that a user will face blocking can be computed in different scenarios as a function of the channel index. In addition, we also plotted the bandwidth capacity that is required to assure that the end-to-end IPTV blocking probability, as a function of channel index i, does not exceed a certain level, which can help the system designer to dedicate the proper bandwidth to the IPTV service in order to guarantee a certain level of Quality of Experience to clients.

10.2 P2PTV

We focused on the popular BitTorrent-like Peer-to-Peer TV systems. To understand the mechanisms and performance of this system, we studied and analyzed it by means of measurements and an analytical model.

In the measurement study, an existing popular P2PTV application, called SopCast, was investigated. Its functionality, the traffic characteristics and the different aspects of Quality of Experience (e.g., blocking, video quality, audio-video synchronization, peer synchronization, channel zapping time) were tested and evaluated through objective and subjective measurements.

In the analytical model of this BitTorrent-like P2PTV system, the performance was analyzed in terms of end-to-end blocking probability. Blocking arises when 1) user U cannot find all needed chunks (next 1 second content) from the randomly chosen partners (in a partner group, peers exchange information about which chunks are available at which partner), or 2) when at least one parent (who directly provides chunks to user U) cannot upload his chunks to user U within 1 second due to insufficient bandwidth, or 3) at least one parent leaves during his uploading period. In the case study, our measurement results on SopCast (e.g., parent distribution and bandwidth distribution) were used as input and other parameter values (e.g., the TV channel popularity distribution, the number of TV channels, the number of active users) were chosen to be similar to the IPTV case study in order to have a fair comparison. Provided the appropriate data is available, our formulas allow for similar computations for different cases.

The analytical results showed that the popular TV channels have a much smaller probability to be blocked than the unpopular channels. The P2PTV end-to-end blocking is mainly contributed by the second cause mentioned above. Hence, the limited upload bandwidth of an end user's parent distributed to him mainly causes the blocking. This clearly indicates the importance of the parent selection policy when designing the system. We also observed that a user will face less blocking if the number of available channels is smaller, or if users leave infrequently, while changing the number of users did not significantly affect the blocking probability. In addition, we compared P2PTV and IPTV based on our computations with realistic data inputs. We assessed which technology (IPTV or P2PTV) incurs the lowest end-to-end blocking and in which scenarios. Currently, if the operator decides to transmit television content to its residential users using P2PTV instead of IPTV, all channels will likely face more blocking than before. However, we can predict that when the number of end users increases, this will change. In a P2PTV system, the number of users has little effect on the blocking, while in an IPTV system the blocking would largely increase if the processing capability of the equipment (e.g. DSLAM) cannot scale accordingly. When the total amount of users increases and the amount of DSLAMs and their processing capability remains the same, there will be a point at which the P2PTV system will start to outperform the IPTV system for popular channels, unless the IPTV network is extended accordingly. In our Dutch case study, this cross-over point is around $297,600$ users.

Hence, is P2PTV ready to replace IPTV? Our measurement results show that P2PTV nowadays can provide good quality video to end users, but the large peer lags (i.e. peers watching the same TV channel might not be synchronized, 3 seconds lag for SopCast) and the huge zapping time (average 50 seconds and maximum 200 seconds for SopCast) hamper it to replace IPTV to deliver the TV streaming content.

10.3 CDN

The centrally controlled Content Delivery Network discussed in this thesis integrates the real-time multicast streaming service and the server-client Video-On-Demand service after the end user pauses the stream and chooses to resume the same stream. We developed a performance model to compute the availability of the required service. The probability that the bandwidth for a required service (either the multicast request or the unicast request after the pause) is available is calculated as a function of video popularity, the number of available videos, cache sizes, and bandwidth. Different contributions to the blocking probability, namely multicast request blocking and unicast request blocking, have been analyzed separately. The model is based on a realistic caching hierarchy and we applied it to a MobileTV service. In the experiment, we focused on a Multimedia Broadcast and Multicast Services (MBMS) architecture with our proposed pausing function, for which we obtained the configuration related parameter values from a Dutch Telco and UMTS network and the user behavior related parameter values from measurement study in [37].

Based on the results in our realistic MobileTV use case, we found that the users' requests are less likely blocked if there are less available TV channels. The results also indicate that the number of TV channels affects the overall quality very much. In order to support more MobileTV channels, while maintaining a similar performance (an overall end-to-end blocking probability less than 1%), increasing the bandwidth between RNC and the Base Station can bring significant improvement. Moreover, in order to experience a good performance after the pause (e.g. with unicast service availability more than 99.99%), the end user only needs a moderately sized local cache (e.g. 31 megabytes in our case) at their Mobile Phone for this service.

Our model allows to compute how many MobileTV channels can be supported for the overall blocking probability not to exceed a certain threshold. Furthermore, we computed the required local cache size to assure that an end-user can successfully resume the streaming with a certain probability after the pause. Our results can be used not only to analyze the blocking in existing stream caching systems, but also to predict the system behavior in the future when the link capacity increases and correspondingly more HDTV channels can be supported.

10.4 P2PVoD

A Video-on-Demand system using Peer-to-Peer technology was analyzed in this thesis. An analytical fluid model was proposed for a mesh-based P2PVoD system and we compared the model to a realistic P2PVoD simulator. This model, which is based on current P2PVoD applications, leads to non-linear system equations, in which the content stored at the end users is released as soon as they complete viewing the video. In a non-linear model, small perturbations of the input may lead to large (undesirable) changes in the behavior of the system. Consequently, we have provided rules for a P2PVoD application that ascertain a linear behavior, where a peer stops sharing his content a fixed amount of time after his download finishes. A critical factor is the seed definite leaving rate. In our model, four phases of the system were analyzed seperately: 1) Start-up phase, 2) Seed Appearance phase, 3) Seed Departure phase, and 4) Steady state. The results in the experiment indicated that our analytical results and simulation results match well.

However, we still observed some limitations in such a system. That is, we have to set the video playback rate v smaller than the average upload rate bw_{up} at a user, in order to have a good system performance *all the time*. If $v > bw_{up}$, we will not only have a so-called start-up phase problem, but there will also be too much burden on the peers. To solve these problems, we proposed an improved system with helpers. With our analysis of this new system, we deduced rules for setting some key parameters (e.g., the new value of the seed serving time, the helper seed serving time, and the helper arrival rate) to make sure that the average download rate of a user is always larger than the video playback rate. In order to optimize the upload bandwidth resource usage, strategies for helpers were also proposed.

With our model, parameters that affect the system performance were observed, and the system stability was investigated. Our model leads to desgin rules for achieving a good and stable system performance all the time under all conditions, while optimizing the system resource usage as well.

10.5 VC

Existing Video Conferencing applications were classified according to their: different terminals, user interfaces, signaling protocols, and network structures. Based on our survey of 18 popular VC applications, we observed that the maximum frame rate is 30 frames/second, which corresponds to regular TV quality. All applications support only a very limited number of participants and the applications that support more than 10 simultaneous participants all use a centralized server-client (S/C) network structure. We further chose four video conferencing applications (*Mebeam*, *Nefsis*, *Vsee*, *Qnext*) for in-depth study, because each represents one of the four architectures (web-based S/C, S/C, decentralized P2P, centralized P2P) under which all 18 applications could be classified. We examined their behavior and mechanisms, and investigated their login process, the call establishment process, the packet size distribution, transfer protocols, traffic load, delivery topology, and different aspects of Quality of Experience. Our experiments were set up both locally and globaly. The global experiments were conducted under similar and stable conditions, with 4 participants involved (2 clients in the Netherlands and 2 clients in China).

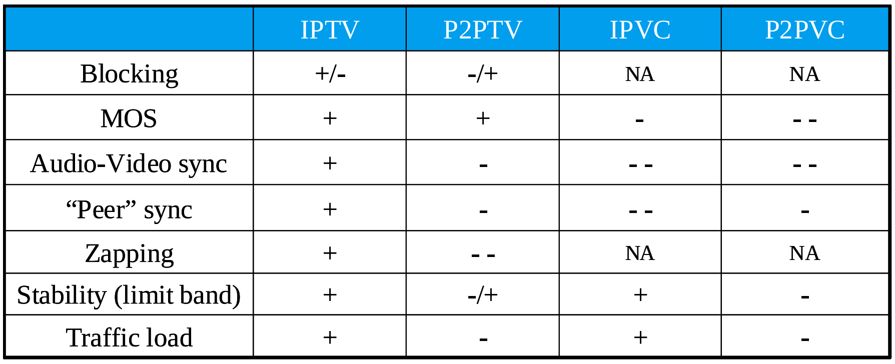
Our main conclusions from the measurement results on the traffic characteristics of four different video conferencing systems are: (1) The QoE of multi-party video conferencing is very sensitive to bandwidth fluctuations, especially in the uplink. Hence, an adaptive resolution/frame rate policy should be deployed; (2) When the number of participants increases, the traffic load at each participant does not always increase correspondingly, suggesting that re-encoding at the video or a traffic shaping policy takes place to control the overall traffic in the system. Our QoE measurement results are summarized as: (1) Compared to non-interactive multimedia services (e.g. P2PTV), existing Internet video conferencing applications in general cannot provide very good quality to their end users (poor video and audio quality, large audio-video lag, and long communication delay in some cases); (2) Only a limited number of multimedia participants are supported and rarely high definition webcamera streaming is supported due to the limited available bandwidth or the limited processing capability; (3) The existing systems are not always reliable. When the network is unstable or the available upload bandwidth is very limited (thresholds have been found), none of the applications work properly.

Based on our observations and analysis, we recommend to incorporate the following aspects when designing video conferencing applications: It seems that the Server-to-Client architecture with many servers located all over the world is currently the best architecture for providing video conferencing via the Internet, because it introduces the least congestion at both servers and clients. Load balancing and load control algorithms help the overall performance of the system and the codec used is important for the quality that end users perceive. The bottleneck to support video conferencing with more participants and high-definition streams is the overhead traffic generated by them. To support more simultaneous participants in a single conferencing session, the traffic load has to be controlled/limited by using traffic shaping or re-encoding the video streams.

Chapter 11

General Conclusion on “To Peer or Not to Peer”

In this Section, we will give a general comparison between IPTV and P2PTV; IPVC and P2PVC from aspects of 1) Blocking, 2) MOS (Mean Opinion Score), 3) Audio-Video Synchronization, 4) User synchronization (Interactivity), 5) Channel zapping time, 6) System stability when the bandwidth is limited, and 7) Resource usage (Traffic Load). In Figure 11.1, “+” represents good performance, while “-” represents bad performance, and “- -” represents worse performance. “+/-” represents good performance right now, but possibly bad performance in the future.



	IPTV	P2PTV	IPVC	P2PVC
Blocking	+/-	-/+	NA	NA
MOS	+	+	-	- -
Audio-Video sync	+	-	- -	- -
“Peer” sync	+	-	- -	-
Zapping	+	- -	NA	NA
Stability (limit band)	+	-/+	+	-
Traffic load	+	-	+	-

Figure 11.1: Pro’s and con’s of using P2P or not for different streaming services.

For the different services, we have given different suggestions in this thesis. In general, the Server-Client approach works fine nowadays. However, with more and more new-added services, as well as more and more users and content involved in the system, P2P technology might become better suited. However, based on our observations, the P2P technologies and applications used nowadays still have much room for improvement. We made suggestions for how to improve existing P2P systems, like deploying helpers in the system and using traffic load balance/shaping algorithms to control the overhead traffic.

Chapter 12

Future Work

The future work can mainly lie into three folders:

- Study the correlation between QoS and QoE, basd on which a real-time non-reference performance assessment framework can be proposed. It would further real-time get the feedback from the system performance and the user perceived quality, and reversely aid in the internal workings of protocols in aspects such as peer selection or bandwidth scheduling.
- Extend the analytical model
    1. to involve and analyze more policies (e.g. peer selection policy, rare-first policy, fail-recovery policy, etc.);
    2. to consider more parameters (e.g. bandwidth resource usage efficiency, content availability and latency) and analyze their effect on the performance;
    3. to study the traffic overload and performance at users, servers and ISP routers, when destributing a same streaming content in a same size of network (under same physical configuration) using Server-Client and Peer-to-Peer respectively.
- Implentment the proposed solutions into real application, and validate them with our analysis.

Appendix A

Deductions

A.1 $B_{Engset}(i)$ in IPTV and $B_{Engset}(k)$ in CDN

The IPTV system with K available channels and m admitted channels can be modeled as an $M/M/m/m/K$ queuing system. In this Engset queue with different channel i arrival rate λ_i and channel i leaving rate μ_i, each channel request can arrive in random order. Karvo *et al.* [31] introduced a binomial probability generation function to deduce the probability $\pi_j^{(i)}$ (j positions are occupied by j channels other than channel i in an infinite capacity system) for this system:

$$\varphi_i(z) = \sum_{j=0}^{\infty} \pi_j^{(i)} z^j = \prod_{k=1}^{K} \frac{q_k + p_k z}{q_i + p_i z} \tag{A.1}$$

in which, $q_i = 1 - p_i$ and $p_i = 1 - e^{-\frac{\lambda_i}{\mu_i}}$. $B_{Engset}(i) = \pi_m^{(i)} / \sum_{j=0}^{m} \pi_j^{(i)}$. According to $\pi_j^{(i)} = \frac{1}{j!} \frac{d^j \varphi_i(z)}{dz^j}|_{z=0}$([32, pp.18]) and Eq. (A.1),

$$B_{Engset}(i) = \frac{\frac{1}{m!} \frac{d^m \left[\prod_{k=1}^{K} \frac{q_k + p_k z}{q_i + p_i z}\right]}{dz^m}\Big|_{z=0}}{\sum_{j=0}^{m} \frac{1}{j!} \frac{d^j \left[\prod_{k=1}^{K} \frac{q_k + p_k z}{q_i + p_i z}\right]}{dz^j}\Big|_{z=0}} \tag{A.2}$$

Our computation of $B_{Engset}(k)$ in CDN is equal to the $B_{Engset}(i)$ for IPTV, but with changing the symbol representing the TV channel index from i to k and also some changes on the following parameters: 1) the number of admitted channels left for multicast $m = \max\{1, \frac{L_{LEX}}{v} - N_{TV}(\pi_{U_{ROOT}} + \pi_{U_{REX}} + \pi_{U_{LEX}})\}$, and 2) $\lambda_k = \lambda_{k\_L_{LEX}}$, $u_k = u_{k\_L_{LEX}}$ and $\mu_k = \mu_{k\_L_{LEX}}$ (see Eqs. 5.3 and 5.7).

A.2 How to set the value of v in the start-up phase, without helpers

We know from Section 6.3.5 that

1) When $\frac{L_i v}{bw_{down}} \geq \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i + \theta_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$ (download bandwidth is the constraint), we use Eqs. (6.6) and (6.7) in steady state;

2) When $\frac{L_i v}{bw_{down}} < \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i + \theta_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$ (upload bandwidth is the constraint), we use Eqs. (6.9) and (6.10) in steady state.

We can know from Fig. 6.2 and 6.3 that the system will perform well when the download bandwidth is fully used, which means that it is better to use Eqs. (6.6) and (6.7) in steady state. It means that the system developer should set parameters such that they satisfy condition 1.

In the *non-linear* system, when assuming $\theta_i = 0$ and ignoring the small B_u, for condition 1, $\frac{L_i v}{bw_{down}} \geq \frac{\lambda_i L_i v - (\frac{\lambda_i bw_{up}}{\gamma_i + \theta_i} + bw_{os})}{\lambda_i bw_{up} + \theta_i bw_{os}}$ is equivalent to $v \leq \frac{1 + \frac{bw_{os}}{\lambda_i L_i bw_{up}}}{\frac{1}{bw_{up}} + \frac{1}{u_i(t)} - \frac{1}{bw_{down}}}$. Then, as long as $v \leq \frac{1}{\frac{1}{bw_{up}} + \frac{1}{u_i(t)} - \frac{1}{bw_{down}}}$, we can use Eqs. (6.6) and (6.7) in *steady state*. Because $u_i(t) = bw_{down}$ when the download bandwidth is fully used, then we only need to set $v \leq bw_{up}$ to satisfy condition 1.

In the *linearized* system, by carefully setting the value of γ_i, we should *always* be able to use Eqs. (6.6) and (6.7) in *steady state*.

However, in the *Start-up phase* ($0 \leq t < \tau_i$), the downloading time $\tau_i = \frac{L_i v}{u_i(t)} \approx \frac{L_i v}{bw_{up}}$ should be smaller than the planned display ending time $L_i + B_u$ in order to make sure that the download finishes before the display ends. To meet this requirement, we need to set $v \leq bw_{up}$ for *both linearized and non-linear* systems.

In other words, the condition of $v \leq bw_{up}$ is needed in both the Start-up phase and the steady state in the non-linear system; but needed only in the Start-up phase in the linearized system. Hence, as long as the system developer makes the video playback rate v smaller than the average upload rate per peer bw_{up}, the system will perform well and stable *all the time*. That is also the reason why most current commercial P2PVoD systems on the Internet provide only videos with SDTV quality (with rates of 0.5 Mbit/s or lower)[1].

On the other hand, no matter how much download capacity bw_{down} is provided, as long as $bw_{down} > v$, it will not affect the end user performance and the system stability when $v \leq bw_{up}$.

---

[1] This is also indicated by the measurement study in [61, Fig 5(c) and Table II], where they found that even the "naive" random pull scheduling (the simplest system design) can perform well when the playback rate is 0.48 Mbit/s (which is smaller than the preset average upload rate of 0.55 Mbit/s).

A.3 Deduction of γ_i in the system with helpers

For the steady state of the system with helpers, we can solve system equations (7.1) to (7.5) if $bw_{down}\bar{x}_i + bw^h_{down}\bar{x}^h_i \leq bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os} + bw^h_{up}(\bar{x}^h_i + \bar{y}^h_i)$

$$\bar{x}_i = \frac{\lambda_i L_i v - \bar{x}^h_i(bw^h_{down} - u^h_i(t))}{\theta_i L_i v + bw_{down}}$$
$$\bar{y}_i = \frac{\lambda_i bw_{down} + \bar{x}^h_i(bw^h_{down} - u^h_i(t))\theta_i}{\gamma_i(\theta_i L_i v + bw_{down})}$$

where $\bar{x}^h_i = \frac{\lambda^h_i C_h}{\theta^h_i C_h + u^h_i(t)}$ and $\bar{y}^h_i = \frac{u^h_i(t)\lambda^h_i}{\gamma^h_i(\theta^h_i C_h + u^h_i(t))}$.

Assuming $\theta_i = 0$ and with the expressions of $\bar{x}_i$ and $\bar{y}_i$, the condition that $bw_{down}\bar{x}_i + bw^h_{down}\bar{x}^h_i \leq bw_{up}(\bar{x}_i + \bar{y}_i) + bw_{os} + bw^h_{up}(\bar{x}^h_i + \bar{y}^h_i)$ amounts to

$$\frac{1}{bw_{down}} \geq \frac{\lambda_i L_i v + \bar{x}^h_i(u^h_i(t) - bw^h_{up}) - (\frac{\lambda_i bw_{up}}{\gamma_i} + bw_{os} + bw^h_{up}\bar{y}^h_i)}{bw_{up}[\lambda_i L_i v - \bar{x}^h_i(bw^h_{down} - u^h_i(t))]} \tag{A.3}$$

We input the values of $\bar{x}^h_i$ and $\bar{y}^h_i$ into Eq. (A.3) and because $\lambda_i L_i v > \bar{x}^h_i(bw^h_{down} - u^h_i(t))$ (which is equivalent to $\lambda_i L_i v > \lambda^h_i C_h(\frac{bw^h_{down}}{u^h_i(t)} - 1)$, where θ^h_i is assumed to be 0 and $u^h_i(t) = bw^h_{down}$ can be set in steady state), we can get the seed serving time $\frac{1}{\gamma_i}$ in the system with helpers, as expressed in Eq. (7.9).

Appendix B

Strategy and Protocols with helpers

The main idea of adding helpers is to bring more upload bandwidth resources to the system. Then, in order to optimize helpers' upload bandwidth usage, we design the protocols for helpers (e.g. how much and what a helper should download). This work is done together with Hao Zhang.

The video content provider maintains a tracker to keep track of all the participating normal peers and helpers in the streaming session and to assist building the overlay network. The normal peers maintain connections to a number of helpers to make up for the streaming rate that cannot be sustained by its peer neighbors. We make sure that an average λ_i^h number of helpers per second come into the system to help the system, starting from downloading the C_h amount of data. After finishing the download, the helper then continues to stay for $\frac{1}{\gamma_i^h}$ seconds as a helper seed before its departure.

In situations of fluctuating demand, it is advantageous that helpers spread out their download chunks into different segments of the video. In this way, a helper can maximize the number of normal peers that potentially need its assistance while keeping the download and storage amount low. For this reason, we divide the helper's download C_h into m segments with one and only one chunk per segment.

The whole video is broken into continuous blocks B_i, $i = 1, 2, ..., N$, each consisting of m segments of equal duration. Each segment contains k chunks, where "chunk" is the minimum transmission unit on P2PVoD network overlay. The helpers form N separate swarms, S_i, with $i = 1, 2, ..., N$, each having at least k helpers responsible for serving one corresponding block B_i. The new incoming normal peer will obtain from the tracker a number of helpers within swarm S_1. As the normal peer continues to watch the video, its playback time will eventually fall beyond what S_1 can supply. Then, the normal peer queries the tracker for a new list of helpers in the next swarm. The normal peer continues to perform similar queries as it keeps streaming until the end of the video. The incoming helpers first join swarm S_1 to serve block B_1. When the number of helpers in S_1 reaches k, the new incoming helpers will then join swarm $S_{2,}$ and so on. In case that some helper seeds depart from the system, the new coming helpers will replace their place in S_1 with the highest priority. If the number of helpers in the system is more than $N \times k$, the redundant helpers can download the rarest or most requested chunks. When a helper fails, the tracker will acquire a new helper to download and store an equivalence of the lost chunks by connecting to a number of available normal peers or those redundancy helpers who already have the corresponding data.

Based on this design guideline, we present the architecture for the helper network as is demonstrated in Figure B.1.

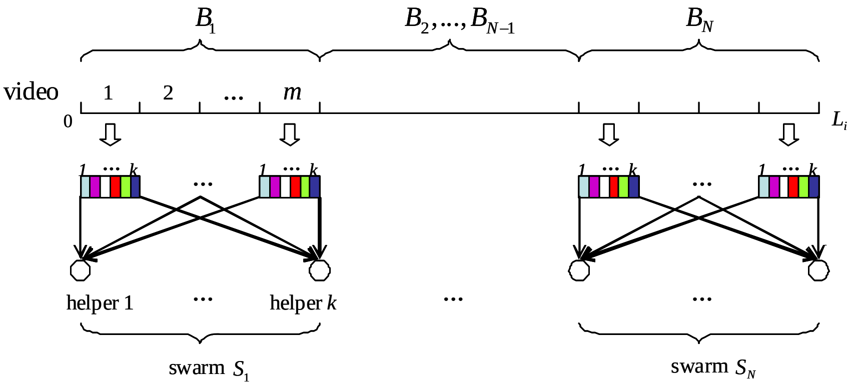


Figure B.1: Helpers' strategy. Helpers break into N swarms each serving one block of video. Each swarm consists of at least k helpers each carring one unique parity chunk for every segment of the block.

Each helper downloads and stores one and only one parity chunk per segment, for every segment in the corresponding block as is shown in Figure B.1. All the parity chunks the helpers carry are mutually exclusive. In this way, a normal peer can connect to the k helpers in the corresponding swarm to download the content of a corresponding video block. This helps mitigate complex load balancing strategies that are usually needed in such a decentralized system. If a chunk size[1] is w, then we have $C_h = m \times w$.

In general, a helper behaves like a normal peer, but only downloads the appointed specific C_h amount of video content instead of the whole L_i length of video.

---

[1] The chunk size of SopCast stream is 10 Kbytes, approximately 5 chunks/second; but the chunk size may differ depending on the application.

Appendix C

Notations

C.1 Notation P2PTV model

v :	The playback rate of the channel i (kbits/s).
R :	The number of uniform size chunks in one second of TV content.
U :	A random user.
$\mathcal{P}$:	A partner group of user U for the channel i.
M :	The number of parents of user U, chosen from $\mathcal{P}$.
Y :	The number of children of user U.
bw_{up} :	The upload bandwidth of a parent.
i :	The TV channel index.
$b_{chunk}(i)$:	The probability that the required R chunks cannot all be found in $\mathcal{P}$.
$b_{time}(i)$:	The probability that at least one parent u cannot upload his chunks to user U within 1 second due to insufficient bandwidth, for channel i.
$b_{dyn}(i)$:	The probability that $\geqslant 1$ parent leaves during his uploading period.
$b(i)$:	The end-to-end blocking probability for channel i.
$B_i(r)$:	The probability that user U cannot find chunk r successfully among $\lvert\mathcal{P}\rvert$ randomly chosen partners, for channel i.
$\pi_i(r)$:	The probability that a peer is storing chunk r, for channel i.
$bw[u,U]$:	The upload bandwidth that a parent u has available for user U.
$\Pr[bw[u,U] \leq x]$:	A distribution function of $bw[u,U]$.
X_u :	The number of chunks the user U requests from parent u.
$p_M(R)$:	The number of partitions of R into parts not exceeding M.
$\Pr[M=k]$:	Density function of the occurrence probability of having M parents.
$X_u(j)$:	How many chunks user U requests from parent u.
$Z(t)$:	The peer departure process.
θ_i :	The rate at which peers leave from channel i.
$P_u(j)$:	The probability that parent u with $X_u(j)$ chunks (in partition j) leaves during uploading.
N :	The number of concurrent active peers over all channels.

α_i : The popularity of channel i.
N_i : The total number of available peers in channel i, equal to $N\alpha_i$.
λ : The arrival rate of users into the P2PTV system. $Q_{P2PTV} = \frac{\lambda}{\theta_{i.}}$.
p_i : The probability that channel i is "on".
q_i : The probability that channel i is "off".
bw_{X_u}: The required upload bandwidth for parent u to deliver X_u chunks to U in time.
$\pi_j^{(i)}$: The probability that j positions are occupied by j channels other than channel i in an infinite capacity system.
$\varphi_i(z)$: A binomial probability generation function to deduce $\pi_j^{(i)}$.

C.2 Notation IPTV model

C : The capacity of a link from a particular DSLAM to the edge router.
m : The maximum number of channels that can be transmitted simultaneously over a link with capacity C, equal to $\left\lfloor \frac{C}{C_o} \right\rfloor$.
C_o The capacity of one television channel.
K : The available television channels that can be viewed.
α_i : The popularity of TV channel i, where $0 \leq \alpha_i \leq 1$.
$\lambda(t)$: The request arrival rate at time t.
B_{proc} : The blocking caused by the limited processing capability of a DSLAM.
$B_{link}(i)$: The blocking probability of TV channel i, caused by the insufficient available capacity from the DSLAM to the edge router.
$B(i)$: The end-to-end blocking probability for channel i. In another word, the probability that the request of channel i is blocked.
s : The number of users accessing the DSLAM.
n : The number of replications that the DSLAM can handle.
λ_{DSLAM} : The rate at which a user requests a TV service.
μ_{DSLAM} : The rate at which the user turns his TV off. $\rho_{DSLAM} = \frac{\lambda_{DSLAM}}{\mu_{DSLAM}}$
$P(i)$: The probability that channel i is "on".
$B_{Engset}(i)$: The probability that the link from the DSLAM to the edge router is consumed by m channels other than the requested channel i.
λ : The users' arrival rate in the IPTV system, includes both the rate users switch on their television as well as the channel switching rate.
λ_i : The users' arrival rate in channel i, equal to $\alpha_i\lambda$
u_i : The number of users leaving from channel i per second. $\rho_i = \lambda_i/u_i$
Q_{IPTV} $= \lambda/u_i$

C.3 Notation CDN model

$BM\text{-}SC$:	The highest level of a caching hierarchy, as a MobileTV content data center.
GSN :	The 2nd level of a caching hierarchy with Regional Exchanges.
RNC :	The 3rd level of a caching hierarchy with Local Exchanges.
BS :	Base Stations which will not be equipped with a cache.
UE :	User Equipment (i.e. Mobile Phone) at the lowest level of the hierarchy.
L_i :	The capacity (Mbit/s) at level i available for the video streaming service, where $i \in \{BS, RNC, GSN, BM - SC\}$.
v :	The video streaming rate (Mbit/s).
N_{TV} :	The average number of registered TV users attaching to one BS.
λ_{TV} :	How many users attaching to one BS turn on their TV every second.
α_k :	The popularity of video channel k.
$1/\mu_s$:	The time length the user is viewing a video before zapping to a different one.
$1/\mu_v$:	The period a user watches the video stream before pausing.
$1/\mu_L$:	The period a user continuously watches the video stream before leaving.
$1/\mu_p$:	The duration of the pause.
P_s :	Immediately after the pause, the probability that the end user chooses to switch to another multicast video.
P_L :	Immediately after the pause, the probability that the end user chooses to turn off the TV and leave from the system.
P_i :	The probability that the stream is unicast from cache at level i after a pause.
T_i :	The capacity of a cache at level i reserved for video streams (in seconds).
C_i :	The amount of cache disk space (in bits) at level i reserved for caching video streams, where $i \in \{UE, RNC, GSN, BM - SC\}$.

π_{U_i} :	The probability that a user who registered for a video streaming service is in state "U_i" (the unicast state where the user resumes the video from cache i after having paused) in steady state; $i \in \{UE, RNC, GSN, BM - SC\}$.
π_{off} :	The probability that a user is in state "off" (the offline state where the user is not using the service) in steady state.
π_{Pause}:	The probability that a user is in state "$Pause$" (the state where the user is pausing) in steady state.
π_M :	The probability that a user is in state "M" (where the user has started viewing the video and it is delivered via multicast) in steady state.
$B(k)$:	The probability that a user cannot get access to the service of his choice.
$B_m(k)$:	The end-to-end (E2E) multicast request blocking for the stream k.
$B_{u;i}$:	The E2E unicast request blocking if the user retrieves the delayed video data from cache i after the pause.
$E[B_{u;i}]$:	The mean unicast request blocking probability.

$B_{u\_L_i}$: The mean probability that a unicast request is blocked at level i.

$\pi_{j\_L_i}$: The probability that j video streams are multicast at L_i in steady state.

$B_{u\_L_i\_j}$: The blocking probability of a unicast request at link L_i when j video streams are multicast at link L_i.

$\varphi(z)$: The binomial probability generation function of $\pi_{j\_L_i}$.

$p_{k\_L_i}$: The probability the multicast channel k is "on" at L_i. $q_{k\_L_i} = 1 - p_{k\_L_i}$.

$\lambda_{k\_L_i}$: The users' multicast requests arrival rate at video channel k in link L_i.

$u_{k\_L_i}$: Channel k users' leaving rate at link L_i.

$\mu_{k\_L_i}$: The leaving rate of channel k from link L_i.

$P(k)$: The probability that channel k is "on".

K : The number of available multicast TV channels.

$B_{Engset}(k)$: The probability that the link L_{RNC} is consumed by m multicast channels other than the requested channel k.

λ_k : Channel k's multicast requests arrival rate in the model, which is equal to the channel k's arrival rate in link L_{RNC}, $\lambda_{k\_L_{RNC}}$.

u_k : Multicast users' leaving rate from channel k in the model, which is equal to the leaving rate of users from link L_{RNC}, $u_{k\_L_{RNC}}$. $\rho_k = \lambda_k/u_k$.

μ_k : Channel k's leaving rate from channel k in the multicast blocking model, which is equal to the leaving rate of channel k from link L_{RNC}, $\mu_{k\_L_{RNC}}$.

$L_{i\_j}^{(unicast)}$: The bandwidth for unicast when there are j multicast videos at level i.

n_{i_j} : The available unicast servers at level i.

λ_i : The arrival rate of unicast requests at link L_i.

β_i : The leaving rate of unicast transmissions at link L_i.

C.4 Notation P2PVoD model

$\eta_i(t)$: The probability that a downloader is sharing his content with others.
$T_i(t)$: The downloading time of the whole video file i for a peer at time t.
$x_i(t)$: The number of downloaders in video i system at t. We denote $\bar{x}_i = \lim\limits_{t \to \infty} x_i(t)$.
$y_i(t)$: The number of seeds in the video i system at t. We denote $\bar{y}_i = \lim\limits_{t \to \infty} y_i(t)$.
$x_i^h(t)$: The number of helper downloaders in the video i system at t. $\bar{x}_i^h = \lim\limits_{t \to \infty} x_i^h(t)$.
$y_i^h(t)$: The number of helper seeds in the video i system at t. $\bar{y}_i^h = \lim\limits_{t \to \infty} y_i^h(t)$.
bw_{up}^h : The average upload rate at a helper for video delivery.
θ_i^h : Helper's random leaving rate from video i.
$u_i^h(t)$: The average download rate at each helper at time t for video i.
bw_{down}^h : The average download capacity (the maximum download rate) per helper.
bw_{os} : The upload bandwidth of the original seed.
θ_i : The random leaving rate of a peer from video i.
λ_i : The peers' arrival rate to video i.
γ_i : The constant seed leaving rate from video i, set in the linearized system.
$\gamma_i(t)$: The seed leaving rate at time t in the non-linear system.
τ_i : The time to complete downloading video i for the first seed.
$u_i(t)$: The average download rate at each peer at time t for video i.
B_u : The prebuffer size at each peer.
bw_{down} : The average download capacity per peer.
bw_{up} : The average upload rate per peer.
λ_i^h : The avg. number of helpers coming to help the system per second.
$\frac{1}{\gamma_i^h}$: The avg. time a helper seed stays in the system after finishing its download.
v : The streaming rate of the video.
N : The number of blocks (referred to as B_i) in the whole video.
The number of swarms (referred to as S_i) for helpers. $(i = 1, 2, ..., N)$
m : The number of segments per block. No. of segments each helper downloads.
k : The number of chunks per segment. The number of helpers per swarm.
w : The size of a chunk.
C_h : The amount of video data a helper needs to download.
L_i : The length of the video i in VoD system.

Appendix D

Abbreviations

IPTV : IP multicast Television
P2PTV : Peer-to-Peer Television
CDN : Content Delivery Networking
VoD : Video-on-Demand
IPVC : IP (Server-to-Client) Video Conferencing
P2PVC : Peer-to-Peer Video Conferencing
VC : Video Conferencing
QoE : Quality of Experience
QoS : Quality of Service
E2E : End-to-End
MOS : Mean Opinion Score
bVQM : batch Video Quality Metrics
DSLAM : Digital Subscriber Line Access Multiplexer
ITU : The International Telecommunications Union

Bibliography

[1] Bram Cohen, Bittorrent protocol 1.0, www.bittorrent.org, 2002.

[2] X. Hei, C. Liang, J. Liang, Y. Liu and K. W. Ross, "A Measurement Study of a large-Scale P2P IPTV System", IEEE Transactions on Multimedia, vol. 9, no. 8, pp. 1672-1687, 2007.

[3] X. Zhang, J. Liu, B. Li, and TS. P. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media streaming", Proc. of IEEE INFOCOM, Mar, 2005, vol.3, pp.2102-2111.

[4] J. Wang, "A survey of Web caching schemes for the Internet", ACM SIGCOMM CCR, vol. 29, pp. 36-46, October, 1999.

[5] F.T.H. den Hartog, B.L.G. Bastiaans, M.A. Blom, M.G.M. Pluijmaekers, R.D. van der Mei, "The use of Residential Gateways in Content Delivery Networking", ATNAC, Sydney, Australia, December, 2004.

[6] F.T.H. den Hartog, N.H.G. Baken, D.V. Keyson, J.J.B Kwaaitaal, and W.A.M. Snijders, "Tackling the complexity of Residential Gateways in an unbundling value chain", ISSLS, Edinburg, UK, 2004.

[7] Ira M. Weinstein, "Making the Best of ISDN-Based Videoconferencing", Wainhouse Research, 2004.

[8] The International Telecommunication Union (ITU) E 10669 (11/98) Recommendation H.323.

[9] Design Guide for the Cisco Unified Videoconferencing Solution Using Desktop Component Release 7.0 © 2009 Cisco Systems, Inc., December 2009.

[10] Network Working Group, RFC 3261 AT&T, 2002.

[11] The International Telecommunication Union (ITU), ISO/IEC JTC1/SC29/WG11 N3536, Beijing, 2000.

[12] B. Fallica, Y. Lu, F.A. Kuipers, R. Kooij, and P. Van Mieghem, "On the Quality of Experience of SopCast", Proc. of IEEE Future Multimedia Networking (FMN'08), Cardiff, Wales, UK, September 17-18, 2008.

[13] T. Silverston, O. Fourmaux, "P2P IPTV Measurement: A Case Study of TVAnts", in procedings of student workshop of Conference on Future Networking Technologies (CONEXT'06), December 2006.

[14] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M. van Steen, H.J. Sips, "Tribler: A social based Peer to Peer system," Proc. of IPTPS, Feb. 27-28, 2006.

[15] S. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming", In proc. of ICST Workshop on Recent Advances in Peer-to-Peer streaming, 2006.

[16] T. Silverston, O. Fourmaux, "Measuring P2P IPTV Systems", in proceedings of Network & Operating Systems Support for Digital Audio & Video (NOSSDAV'07), June 2007.

[17] S. Tang, Y. Lu, J. Martin Hernandez, F.A. Kuipers, and P. Van Mieghem, "Topology dynamics in a P2PTV network", Proc. of IFIP Networking 2009, Germany, May 11-15, 2009.

[18] A. Orebaugh, G. Ramirez, J. Burke, and L. Pesce, "Wireshark & ethereal network protocol analyzer toolkit (jay beale's open source security)", Syngress Publishing, 2006.

[19] Y. Lu, F.A. Kuipers, M. Janic, and P. Van Mieghem, "E2E blocking probability of IPTV and P2PTV," Proc. of IFIP Networking 2008, Singpore, May, 2008.

[20] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, S. Tewari, "Will IPTV ride the peer-to-peer stream?", Communications Magazine, IEEE, Volume: 45, Issue: 6, On page(s): 86-92, June 2007.

[21] ITU-T Rec. P.800, "Methods for Subjective Determination of Transmission Quality", 1996.

[22] M.H. Pinson and Stephen Wolf, "A New Standardized Method for Objectively Measuring Video Quality", IEEE Transactions on Broadcasting, Vol. 50, No. 3, pp. 312-322, Sep. 2004.

[23] The International Telecommunications Union (ITU) BT.1359-1 (11/98) Relative timing of sound and vision for broadcasting.

[24] DSL Forum, "Triple Play Services Quality of Experience (QoE) Requirements and Mechanisms", Technical Report TR-126, 13 Dec., 2006.

[25] M. Meo and F. Milan, "QoS-aware Content Management in P2P Networks", Proc. of HOT-P2P'04.

[26] R. Susitaival, S. Aalto, and J. Virtamo, "Analyzing the dynamics and resource usage of P2P file sharing by a spatio-temporal model", Proc. of P2P-HPCS'06.

[27] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding and X. Zhang, "A Performance Study of BitTorrent-like Peer-to-Peer Systems", IEEE Journal on selected areas in communications, vol. 25, no. 1, January 2007.

[28] H. Rademacher, *Topics in Analytic Number Theory*, Springer-Verlag Berlin-Heidelberg-New York, 1973.

[29] C. Vassilakis, N. Laoutaris and I. Stavrakakis, "On the Benefits of Synchronized Playout in Peer-to-Peer Streaming", Proc. of CoNEXT, September 2006.

[30] S.Y. Lim, J.M. Soek and H.K. Lee, "A path control architecture for receiving various multimedia contents", Proc. of ICACT2006, February 2006.

[31] J. Karvo, J. Virtamo, S. Aalto, O. Martikainen, "Blocking of dynamic multicast connections in a single link", Proc. of IEEE BROADNETS, 1998.

[32] P. Van Mieghem, *Performance Analysis of communications Networks and Systems*, Cambridge University Press, 2006.

[33] Y. Lu, F.A. Kuipers, F. den Hartog, P. Van Mieghem, "Blocking probability of streaming services in a Content Distribution Network", submitted to CCNC, 2010.

[34] P. Rodriguez, C. Spanner, and E.W. Biersack, "Web caching architectures: hierarchical and distributed caching", 4th International Web Caching Workshop, San Diego, 1999.

[35] 3GPP, MBMS Architecture and Functional Description, Technical Specification, TS 23.246, Release 6, June 2006.

[36] A. Pitsillides and C. Christophorou, "MBMS Handover control: A new approach for efficient handover in MBMS enabled 3G cellular networks", Computer Networks, 51(18):4897-4918, 2007.

[37] J. Holub, J. G. Beerend, and R. Smid, "A Dependence between Average Call Duration and Voice Transmission Quality: Measurement and Applications", Wireless Telecommunications Symposium, Pomona, California, May, 2004.

[38] "Mobile TV | iLiveTV, Mobistar Case Study", © Copyright 2010 Envivio, January 2010.

[39] Y. Lu, J.D.D. Mol, F.A. Kuipers, P. Van Mieghem, "Analytical Model for Mesh-based P2PVoD", the 10th IEEE International Symposium on Multimedia (ISM2008), Berkeley, California, USA, December 15-17, 2008.

[40] D. Qiu and S. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer networks," Proc. of ACM SIGCOMM 2004, August 2004.

[41] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," Proc. of IEEE INFOCOM 2007.

[42] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting Streaming Applications," Proc. of IEEE INFOCOM 2006.

[43] J.J.D. Mol, J.A. Pouwelse, M. Meulpolder, D.H.J. Epema, and H.J. Sips, "Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems", Proc. of SPIE, MMCN 2008.

[44] H. Chi, Q. Zhang, J. Jia and X. Shen, "Efficient Search and Scheduling in P2P-based Media-on-Demand Streaming Service", IEEE Journal on selected areas in communications, VOL.25, NO.1, January 2007.

[45] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," in Journal of Peer-to-Peer Networking and Applications, by Springer New York, Feburary, 2008.

[46] G. Arfken, "Eigenvectors, Eigenvalues.", in Mathematical Methods for Physicists, 3rd ed. Orlando, FL: Academic Press, pp. 229-237, 1985.

[47] J. Wong, "Enhancing collaborative content delivery with helpers," Master's thesis, Univeristy of British Columbia, Nov 2004.

[48] J. Wang, C. Yeo, V. Prabhakaran, and K. Ramchandran, "On the role of helpers in peer-to-peer file download systems: design, analysis and simulation", International Workshop on Peer-to-Peer Systems (IPTPS), February 2007.

[49] H. Schulzrinne and J. Rosenberg," A Comparison of SIP and H.323 for Internet Telephony", Network and Operating System Support for Digit Audio and Video (NOSSDAV), Cambridge, England, Jul. 1998.

[50] J.M. Ho, J.C. Hu, and P. Steenkiste, "A conference gateway supporting interoperability between SIP and H.323", Proceedings of the ninth ACM international conference on Multimedia, Ottawa, Canada, 2001.

[51] Y. Lu, Y. Zhao, F.A. Kuipers, and P. Van Mieghem, "Measurement Study of Multi-Party Video conferencing", IFIP Networking, Chennai, India, 2010.

[52] S.A. Baset and H. Schulzrinne, “An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol”, INFOCOM’06, Barcelona, Spain, April, 2006.

[53] L. De Cicco, S. Mascolo, and V. Palmisano, “Skype Video Responsiveness to Bandwidth Variations”, NOSSDAV’08, Braunschweig, Germany, May, 2008.

[54] R. Spiers and N. Ventura, “An Evaluation of Architectures for IMS Based Video Conferencing”, Technical Report of University of Cape Town, 2008.

[55] M.S. Silver, “Browser-based applications: popular but flawed?”, Information Systems and E-Business Management, Vol. 4, No. 4, October, 2006.

[56] G. Trueb, S. Lammers, and P. Calyam, “High Definition Videoconferencing: Codec Performance, Security, and Collaboration Tools”, REU Report, Ohio Supercomputer Center, USA, 2007.

[57] A.W. Rix, “A new PESQ-LQ scale to assist comparison between P.862 PESQ score and subjective MOS”, ITU-T SG12 COM12-D86, May, 2003.

[58] Y. Lu, B. Fallica, F.A. Kuipers, R. Kooij, and P. Van Mieghem,“Assessing the Quality of Experience of SopCast”, International Journal of Internet Protocol Technology, Vol. 4, No. 1, pp. 11-23, 2009.

[59] J.L. Lias. “HDMI’s Lip Sync and audio-video synchronization for broadcast and home video”, Simplay Labs, LLC, August, 2008.

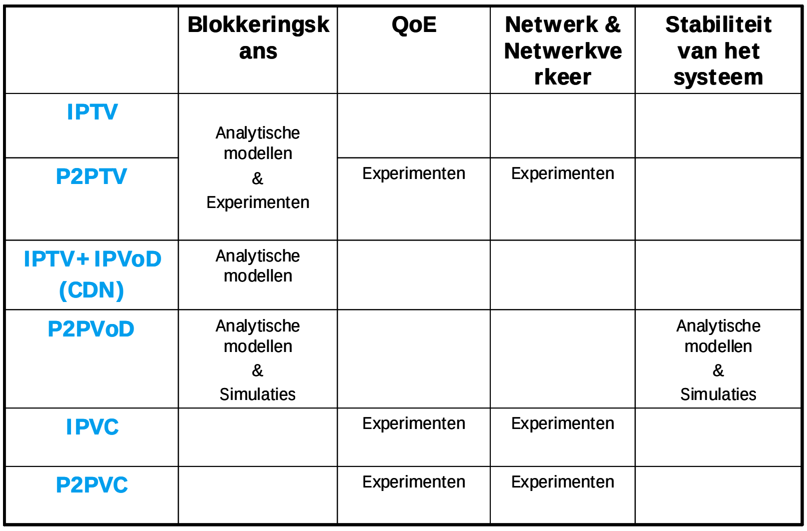
[60] I. Bartoli, G. Iacovoni, and F. Ubaldi, “A synchronization control scheme for Videoconferencing services”, Journal of multimedia, Vol. 2, No. 4, August, 2007.

[61] C. Liang, Y. Guo, and Y. Liu, “Investigating the Scheduling Sensitivity of P2P Video Streaming: An Experimental Study”, IEEE Transactions on Multimedia, Vol. 11, No. 3, pp. 348-360, April 2009.

Samenvatting (Summary in Dutch)

Het aantal multimedia diensten is met een enorme snelheid toegenomen in de afgelopen jaren. Een groot aantal van deze 'streaming' multimedia systemen zijn ingevoerd op de consumentenmarkt. Internet Service Providers, Telecom Operators, Service/Content leveranciers, en eindgebruikers zijn geïnteresseerd in hoe die systemen werken, hoe de gebruiker hun kwaliteit (de Quality-of-Experience (QoE)) ervaart (bijv. met betrekking tot audio / video kwaliteit, audio / video-synchronisatie, vertraging, benodigde tijd voor het opstarten, enz.), de middelen die ze nodig hebben, de stabiliteit van het systeem, en de beschikbaarheid van de service. De streaming multimedia systemen die in dit proefschrift geanalyseerd worden zijn 'IP-multicast TV (IPTV)', 'Peer-to-Peer TV (P2PTV)', 'Content Delivery Networking (CDN)', 'Peer-to-Peer Video-on-Demand (P2PVoD)', 'Server-to-Client Video Conferencing (IPVC)', en 'Peer-to-Peer Video Conferencing (P2PVC)'. Zie figuur 1 voor een overzicht van het onderzoek zoals beschreven in dit proefschrift.

Dit proefschrift beoogt de verschillende soorten populaire streaming systemen te bestuderen door middel van analytische modellen, experimenten, en simulaties, zodat we hun kenmerken en prestaties in verschillende scenarios beter kunnen begrijpen. Middels dit onderzoek, kunnen we naast het beter begrijpen van het gedrag en de beperkingen van bestaande systemen en het ontdekken van de belangrijkste parameters die hun prestaties beïnvloeden, ook potentiële problemen onderzoeken en voorspellen wat de prestaties van het systeem in toekomstige gevallen zal zijn. Door het vergelijken van de twee algemene streaming methoden (Server-to-Client en Peer-to-Peer), verkrijgen we goed inzicht in 'welke methode is beter' en 'wat bepaalt of iets beter is' voor de verschillende diensten en in verschillende scenario's.



	Blokkeringskans	QoE	Netwerk & Netwerkverkeer	Stabiliteit van het systeem
IPTV	Analytische modellen & Experimenten			
P2PTV		Experimenten	Experimenten	
IPTV+IPVoD (CDN)	Analytische modellen			
P2PVoD	Analytische modellen & Simulaties			Analytische modellen & Simulaties
IPVC		Experimenten	Experimenten	
P2PVC		Experimenten	Experimenten	

Figure D.1: Overzicht van het onderzoek zoals beschreven in dit proefschrift.

Acknowledgements

First of all, I would like to thank my supervisors Dr. Fernando Kuipers and Prof. Piet Van Mieghem for their endless patience and kind help. With their guidance, I can more easily get on track in the early stage of my PhD and I clearly see the improvement of my paper writing and my presentation skills. Besides their inspiring ideas to upgrade my work and countless times of modifying my scientific papers, I also learned from them a serious research attitude and the importance of team work. I am also very grateful to get their friendly help to solve life problems other than work. Here, I would like to particularly thank Fernando for being a careful and patient daily advisor.

Furthermore, during my research it was my honor to have lots of cooperations and have fruitful brainstorms with Rob Kooij, Jan David Mol, Frank den Hartog, Milena Janic, Siyu Tang, Javier Martin Hernandez, Rene Serral Gracia, Hao Zhang, Rogier Noldus, Ad Bresser, Edgar van Boven, Xiaoming Zhou and MSc. students Benny Fallica, Yong Zhao, Zhen Qin. In addition, I would like to thank the Dutch Research Delta (DRD) and the Trans sector Research Academy for complex Networks and Services (TRANS), collaborative efforts between KPN, TNO and several Dutch universities, which funded my PhD research. Besides providing a vibrant enviornment to closely work with industries, DRD also offers PhD students training courses from which I learned and enjoyed a lot. I would like to also thank another project I involved in, CONTENT project, which 3 times a year brings together researchers from 11 European universities and 11 international companies to share inspiring ideas and set up cooperations.

In addition, my 4-year PhD life became colorful and enjoyable, thanks to my dear friends on 19th floor, Fernando, Piet, Edgar, Antonio, Anteneh, Javier, Siyu, Jasmina, Rob, Huijuan, Jing, Gu, Javad, Tom, Rogier, Xiaoming, Xueli, Bingjie, Dajie, Wynand, Stojan, Norbert, Christian, Ebisa, Katja, Nico, Ignas, Jos and all the others. I feel very happy and lucky to have such nice colleagues and at the same time friends. I appreciate them for giving me so much kindness, care, and unforgettable memories during these years. I would like to also thank Marjon, Wendy, Stefanie, Dominique and Laura for their assistance during these years.

Last but not least, my special thanks go to my husband, my mom and dad, and my friends for their generous support, encouragement, and love during my PhD.

Curriculum Vitae

Yue Lu, born on March 6th, 1982, is a PhD student in the Network Architecture and Services (NAS) Group, Department of Telecommunication, Delft University of Technology, the Netherlands. She graduated as a B.Sc. student in Electronics and Information Engineering Department at HUAZHONG University of Science & Technology in July 2004. After she obtained her M.Sc. degree from the Faculty of Electrical Engineering at Delft University of Technology in 2006, she started her PhD research, working on multimedia streaming services and networking performance analysis. Since 2004, she has been researching on P2P systems, real-time Internet streaming services (e.g. TV streaming, Video on Demand, Audio/Video Conferencing), their Quality of Service and Quality of Experience, etc.

During her PhD period, she was nominated for the best paper award in IFIP Networking 2008 conference (4 out of 249 submissions); and won the best paper award in IEEE Future Multimedia Networking workshop in 2008 (1 out of 75); and won the best paper award in IEEE International Symposium on Multimedia conference in 2008 (1 out of 196). She received the Netelcom Award runner-up in 2008, which rewards the most innovative scientific contribution of PhD students active within the Dutch Research Delta. She was the Best Presentation Award finalist in the Dutch Research Delta conference contest, in 2010.

Publications:

- Y. Lu, F.A. Kuipers, M. Janic, and P. Van Mieghem, "E2E blocking probability of IPTV and P2PTV", IFIP Networking 2008, Singapore, May 5-9, 2008 (best paper award runner-up).
- B. Fallica, Y. Lu, F.A. Kuipers, R. Kooij, and P. Van Mieghem, "On the Quality of Experience of SopCast", IEEE Future Multimedia Networking (FMN'08), Cardiff, Wales, UK, September 17-18, 2008 (best paper award).
- Y. Lu, J.D.D. Mol, F.A. Kuipers, and P. Van Mieghem, "Analytical Model for Mesh-based P2PVoD", the 10th IEEE International Symposium on Multimedia (ISM2008), Berkeley, California, USA, December 15-17, 2008 (best paper award).

- Y. Lu, B. Fallica, F.A. Kuipers, R. Kooij, and P. Van Mieghem, “Assessing the Quality of Experience of SopCast”, the International Journal of Internet Protocol Technology, 2009.
- S. Tang, Y. Lu, J.M. Hernández, F.A. Kuipers, and P. Van Mieghem, “Topology dynamics in a P2PTV network”, IFIP Networking, Aachen, Germany, May 11-15, 2009.
- Y. Lu, Y. Zhao, F.A. Kuipers, and P. Van Mieghem, “Measurement Study of Multi-Party Video conferencing”, IFIP Networking, Chennai, India, 2010.
- Y. Lu, F.A. Kuipers, F. den Hartog, and P. Van Mieghem, “Blocking probability in a caching hierarchy network”, accepted, IEEE CCNC, Las Vegas, USA, 2011.
- J.L. Zhou, R.V. Prasad, Y. Lu, and I.G.M.M. Niemegeers, “Analysis of a Multihop Integrated UMTS and WLAN Network”, submitted to Journal Telecommunication Systems, Springer, 2010.
- R.S. Gracia, Y. Lu, M. Yannuzzi, and X.M. Bruin, “Effective quality assessment in P2PTV overlays”, to submit to Journal Computer Networks, 2010.