The Impact of Highly Interactive Workloads on Video-on-Demand Systems

Andrew Brampton B.Sc (Hons)

Thesis submitted for the degree of
Doctor of Philosophy

Computing Department
Lancaster University

September 2008

The Impact of Highly Interactive Workloads on Video-on-Demand Systems

Andrew Brampton B.Sc (Hons)

Thesis submitted for the degree of Doctor of Philosophy

September 2008

Abstract

The traditional start-to-finish playback model is not suitable for all modern interactive video streams. Users expect support for higher levels of interactivity such as fast forward and rewind or the ability to arbitrary seek within their media quickly and efficiently. By conducting user studies we have observed start-to-finish is not applicable to many genres of video, and that different playback models fit better. We discuss how existing delivery techniques are impacted by these new observations.

Novel interactive controls such as bookmarks have also highly impacted user behaviour. This has lead to the segments within the media being accessed in a uneven fashion, causing hotspots of interest to form; areas with orders of magnitudes more viewers than others. These hotspots typically began at the beginning of a bookmark, however not always, which lead us to design a dynamic bookmark positioning algorithm. As well as their position, determining the hotspot's length can be beneficial. This aids in autonomic techniques such as replication and pre-fetching as well as allowing the users to find what they want quicker.

Under high level of interactivity, delivery techniques are less efficient due to the unpredictability of the users. We however developed techniques which restore some of this predictability, allowing clients or servers to predict future actions based on past user actions. These technique proves exceeding useful for pre-fetching which reduces seek latencies for client and can reduce load on servers. However knowledge of past user activities need to be gathered from network, thus we develop techniques to do this in a distributed manner.

Declaration

This thesis is a presentation of my original research work. No part of this thesis has been submitted elsewhere for any other degree or qualification. All work is my own unless otherwise stated. The work was carried out under the guidance of Laurent Mathy and Nicholas Race, at Lancaster University's Computing Department.

	30th September 2008
Andrew Brampton	Date

Acknowledgments

Writing my thesis has been one of the hardest tasks in my life, and I couldn't have completed it without the support from my friends, family and the computing department.

Firstly, I would like to thank my two supervisors, Laurent Mathy and Nicholas Race. They have kept me on the straight and narrow, always pointing me in the right direction. I would like to especially thank Laurent, for his tough supervision, which at times seemed harsh, but on reflection pushed me to become a far better researcher. I am also appreciative of Michael Fry's guidance, who in the last year has infused me with new ideas, and direction.

My friends have been there for me through my university life and constantly offering distractions from my studies, sometimes when I needed it, and others times when I didn't! Nevertheless, their backing has been greatly appreciated, especially my housemates, who have offered a good sounding board when I needed to work through a problem. A special thank you should go to Nicola, who in the last few weeks has greatly helped with proofreading of my thesis.

Finally, I would like to thank my parents and brother for being supportive during my PhD journey, and there for me when I needed them.

"Time waits for no one"

The Rolling Stones - 1974

Related Publications

[1] Andrew Brampton, Andrew MacQuire, Idris Rai, Nicholas J. P. Race, Laurent Mathy, and Michael Fry. Characterising user interactivity for sports video-on-demand. In *International workshop on Network and Operating Systems Support for Digital Audio & Video (NOSSDAV)*, June 2007.

[2] Andrew Brampton, Andrew MacQuire, Michael Fry, Idris Rai, Nicholas J. P. Race, and Laurent Mathy. Characterising and exploiting workloads of highly interactive video-on-demand. *Multimedia Systems Journal*, 2008.

[3] Andrew MacQuire, Andrew Brampton, Michael Fry, Nicholas Race, and Laurent Mathy. A case for hybrid content distribution for interactive video-on-demand. In *International workshop on Future Multimedia Networking (FMN)*, September 2008.

Other Publications

[4] Andrew Brampton, Andrew MacQuire, Idris Rai, Nicholas J. P. Race, and Laurent Mathy. Stealth distributed hash table: Unleashing the real potential of peer-to-peer. In *ACM Conference on Emerging Network Experiments and Technology (CoNEXT) (Student Workshop Session)*, October 2005.

[5] Andrew MacQuire, Andrew Brampton, Idris Rai, and Laurent Mathy. Performance analysis of stealth dht with mobile nodes. In *3rd International Workshop on Mobile Peer-to-Peer Computing (MP2P)*, pages 184–189, March 2006.

[6] Andrew MacQuire, Andrew Brampton, Idris Rai, Nicholas J. P. Race, and Laurent Mathy. Authentication in stealth distributed hash tables. In *32nd Euromicro Conference on Software Engineering and Advanced Applications*, August 2006.

[7] Andrew Brampton, Andrew MacQuire, Idris Rai, Nicholas J. P. Race Race, and Laurent Mathy. Stealth distributed hash table: A robust and flexible super-peered dht. In *2nd Conference on Future Networking Technologies (CoNEXT)*, December 2006.

[8] Idris Rai, Andrew Brampton, Andrew MacQuire, and Laurent Mathy. Performance modelling of peer-to-peer routing. In *4th International Workshop on Peer-to-Peer Systems (HOTP2P)*, March 2007.

Table of Contents

Abstract **i**

Acknowledgments **iii**

Publications **iv**

Table of Contents **vi**

1 Introduction **1**

1.1 Research Contributions 2

1.2 Thesis Structure 3

2 Background & Related Work **5**

2.1 Deployed Video-on-Demand Systems 6

2.1.1 Flash-based Sites 6

2.1.2 BBC iPlayer 9

2.1.3 Personal Video Recorders 11

2.1.4 Networked Set-top Boxes (IPTV) 13

2.1.5 Peer-to-Peer 15

2.1.5.1 Streaming 18

2.1.5.2 Video-on-demand 20

2.1.6 Summary 22

2.2 Characterisations of User Behaviour 24

2.2.1 Video-on-Demand 25

2.2.1.1 Popularity 26

2.2.1.2 Popularity Over Time 29

2.2.1.3 Recommendations 30

2.2.1.4 Session duration 31

2.2.1.5 Interactivity 32

2.2.1.6 Segment Popularity 33
2.2.2 Live Streaming 34
2.2.2.1 Time-of-day 35
2.2.2.2 Session duration 35
2.2.2.3 Interactivity 36
2.2.3 Implications 37

3 **Interactive Video-on-Demand System** **39**
3.1 System Overview 42
3.2 Seekable HTTP Flash 43
3.2.1 Flash Indexing 44
3.2.2 Custom Video Player 45
3.2.3 Server Side Support 45
3.3 Experiment 46

4 **Analysis and Modelling** **48**
4.1 Probability Distribution 49
4.1.1 Fitting 52
4.2 Interactions 53
4.3 Seek Distance 55
4.4 Popularity 57
4.5 Longevity 59
4.6 Session Lengths 61
4.7 Inter-seek Times 61
4.8 Sequence 62
4.9 Hotspot Length 64
4.10 User Behaviour Models 66
4.11 Summary 66

5 **Techniques for Interactivity Support** **69**
5.1 Dynamic Placement of Bookmarks 69
5.2 Predictive Pre-fetching 74
5.2.1 Background 75
5.2.2 Pre-fetching Strategies 77
5.2.3 Pre-fetching Knowledge 79
5.3 Hybrid Delivery 83
5.3.1 Periodic Broadcast and Patching 84
5.3.2 Delivery Methods 86
5.3.2.1 Push – Periodic Broadcast over TBCP 87

5.3.2.2 Pull – Peer-to-Peer 87
5.3.2.3 Hybrid – Periodic Broadcast with P2P Patching 88
5.3.3 Experiment 89
5.3.4 Results 91
5.3.4.1 Varying Peer-to-Peer Usage 92
5.3.4.2 Additional Nodes 95
5.3.4.3 Key Observations 96
5.3.5 Conclusion 97

6 Conclusion **98**
6.1 Thesis Contributions 98
6.2 Future Work 99
6.3 Conclusion 102

Bibliography **104**

List of Figures

2.1 Distribution of media via a content distribution network 8
2.2 Distribution of media in a boardcast personal video recorder network . 11
2.3 Distribution of media in a peer-to-peer network 15
2.4 Probability density functions for Zipf and Pareto distributions 26

3.1 Diagrams of our video-on-demand system 40

4.1 Jumps made by users within two videos 54
4.2 CDF of backward and forward seek distances 56
4.3 Object and segment popularity 57
4.4 Number of viewers at each second of video (each vertical line represents the position of a bookmark) 58
4.5 Bookmark utilisation within all videos over time, following initial usage 59
4.6 CDFs of session lengths and inter-seek times 60
4.7 Sequence diagram for Argentina vs. Serbia and Montenegro depicted as a tree 62
4.8 CDF of link probabilities for all videos 63
4.9 CDFs of wait times 64

5.1 Different scenarios that may induce bookmark movement 70
5.2 Manchester United vs Milan single bookmark position 73
5.3 Reduction in viewing duration due to the algorithm 73
5.4 Various metrics for different pre-fetch schemes versus bookmark length 78
5.5 How zero seek latencies is effected by amount of pre-fetch knowledge . 80
5.6 CDFs of Argentina vs. Serbia and Montenegro link frequencies with varying numbers of links in the table 82
5.7 Delivering a Interactive workload with varied group size 91
5.8 Overall segment utility during delivery with varied group size 92
5.9 Network cost of delivery with varied peer-to-peer size 93

5.10 User satisfaction during delivery with varied peer-to-peer size 93
5.11 Effect of additional nodes on overall network stress 95

List of Tables

2.1 Overview of the most popular video sharing websites 7
2.2 Summary of features available with each video-on-demand system . . . 22

4.1 Interactions observed throughout the experiment 54
4.2 A summary of metrics with their corresponding distributions 65

5.1 Example links table, showing the frequency of seeks from one time to another 80

Chapter 1

Introduction

In recent years, the distribution of multimedia rich content has become increasingly popular via the internet. Websites offer a range of media, from short user generated clips to high definition feature films. This content typically has strict delivery requirements, needed for optimal playback. The requirements typically include high bandwidth connections and demands low latencies and low jitter.

Many techniques are employed by both client and server to ensure smooth delivery and to minimise operational costs. These techniques include (but are not limited to) pre-fetching, tree-based distribution, and deploying full content distribution networks (CDNs). Generally, these techniques assume that the user consumes the content conforming to particular usage models. The most commonly assumed models are the classic *start-to-finish* model and an extension of this, the *start-to-end* model. With the former model, users will start playback at the beginning of the content and continue until the very end, whereas the latter model assumes that the users stop playback before the end.

As such systems evolve, users expect more control over the playback of their content, and thus improved functionality. For example, VCR-like controls are already common; fast-forward, rewind, pause and resume. More novel interactivity controls are beginning to appear, for example, bookmarks, which give the user the ability to seek directly to a point of interest within the content, such as a chapter or an event.

Offering services which provide a high level of interactive control creates new challenges for traditional delivery mechanisms. For example, conventional network and application-level multicast is not suitable for providing interactivity. Conversely, simple client-server mechanisms work well under high interactivity, however, they can not easily scale to offer a large number of users these services. Regardless of delivery, there are additional problems such as delay caused by start-up or seek latency, as well as the unpredictable workload placed upon the servers.

Nevertheless, there are numerous commercial video-on-demand services which offer varying degrees of interactivity. Thus far, these systems use a brute-force approach, deploying large scale CDNs to satisfy the needs of their users. This thesis will explain how these existing deployments work, and highlight their flaws. We will then continue by discussing how these techniques can be improved to support interactivity, as well as develop some new techniques.

1.1 Research Contributions

This thesis presents through experimentation new user behaviour models, more applicable to highly interactive content. These models can aid in simulation and development of new techniques for efficient, quick and cheap delivery of content to the user. The models were derived from data obtained by an experimental Video-on-Demand (VoD) website which we designed and deployed. In addition to generic VCR-like features, this custom built VoD application provided advanced interactivity features such as bookmarking. Over a twelve month period more than 1000 unique users were observed accessing a selection of 88 video files. These videos included the entire 2006 FIFA World Cup and the 2007 Eurovision song contest.

Through detailed analysis of the data, common usage models were characterised, such as object popularity, session duration, and other standard metrics. It was observed that when users were offered additional interactive controls, the content was no longer consumed based on the *start-to-end* model. To aid in characterising this novel user behaviour, additional interactive metrics were developed, which better explained this highly interactive system. These include models for how bookmarks are used, as well as models relating to an emergent property, *hotspots*. These *hotspots* are areas of particular interest within the video in which users often choose to watch (and replay) small segments of the full video, in a complete departure from the classic models. While the behaviour observed may be specific to the content used within the experiments, the results may be of general interest, and relevant to other genres of video with popular highlights (e.g., educational, entertainment, news, *etc.*).

This thesis will discuss how current delivery techniques are not designed to handle such levels of interactivity. Understanding these new models can lead to new techniques to improve the delivery of highly interactive media. For example, the actions of a user may now be predicted based on past users. Also, distributed techniques were developed to detect the location of *hotspots* automatically. Knowing the position of *hotspots* presents new opportunities for caching and replication techniques which did not previously exist with less interactive media. Following from this, new hybrid delivery techniques are explored, which use a combination of established delivery protocols. Hybrid delivery allows for quick, efficient, and cheap delivery of content, while offering the user high levels of interactivity not available with existing delivery systems.

1.2 Thesis Structure

After this chapter, this thesis is organised into five chapters. The chapter immediately following this introductory chapter provides background of existing characteristics models in the areas of live and stored streaming media, and their deficiency in modelling interactive behaviour. The background is continued by explaining how interactive media can be delivery, and the problems with this, and then concluding with a discussion of existing video-on-demand deployments and the problems they face.

Chapter 3 describes the design of a video-on-demand system we used to experiment and evaluate new interactive video concepts. This is followed by Chapter 4, which discusses the results obtained from our experiments, with details on characterising the users behaviour and how these can be modelled for future simulations. The evaluation also contains discussion of the implication these new interactive models have on the design of new systems. Chapter 5 builds on the results obtained, and discusses ideas which can improve the delivery of interactive media for both the consumer and distributer. This includes dynamic bookmark placement, pre-fetching and a hybrid delivery technique.

This thesis is concluded with Chapter 6 which gives a overview of the work presented in this thesis. The conclusion also suggest future directions for this research.

Chapter 2

Background & Related Work

The focus of this thesis is understanding how video-on-demand (VoD) systems operate under highly interactive workloads. Once a firm understanding of this is achieved, then new techniques can be designed to help improve performance in such systems. Therefore, the first section of this chapter aims to explain how currently deployed VoD systems work. Special attention is paid to which interactive features these systems already provide, as well as their limitations in providing advanced interactivity. This covers systems such as the incredibly popular YouTube [You08], serving low quality short video clips via a content distribution network (CDN), to systems such as BBC iPlayer [BBC08], a hybrid peer-to-peer CDN platform offing high quality professional content.

Later, to understand exactly how workloads are analysed, Section 2.2, provides a detailed overview of how existing characterisation studies have try to explain and model user behaviour. Modelling behaviour is very important for designing video-on-demand mechanisms, for example, understanding the popularity of objects helps to make caching and replication decisions. Most of the existing research on characterising user behaviour has either ignored, or not experienced high levels of interactivity. This is counter to the results displayed later in this thesis. As such, the review of previous work is discussed in the context of interactive workloads, where applicable. This helps motivate the experiments evaluated in Chapter 4, as well as providing a solid ground to understand the problems with interactivity.

2.1 Deployed Video-on-Demand Systems

There are multiple systems now in place, which allow users to watch videos when they want, how they want. There is an abundance of content available, ranging from short funny video clips, to long feature movies, and everything in between. This has been driven by incredible demand, causing new video-on-demand (VoD) systems to appear, almost daily, to serve different niches. To keep up with demands, these videos are no longer just made by professionals. Anyone with a cheap camcorder or webcam can become famous for 5 minutes. Wired magazine refers to this as "bite-size bits for high-speed munching" [Mil07].

As VoD is becoming more ubiquitous, users are expecting more features. One such feature is interactivity, the ability to pause, resume, and seek within the content. Many of these VoD systems are offering interactive features, however, each with their own limitations. For example, some can only offer interactivity by forcing the user to download the full video first. If they do allow interactivity via streaming, then the systems seem slow or sluggish.

This section highlights the main classes of VoD applications, and well as how they work. Their interactivity features are discussed in detail, as well as their flaws. At the end of this section, there is a summary of all solutions, giving a quick overview of what is out there, and how it operates.

2.1.1 Flash-based Sites

A big push in storing videos online has been the creation of *user generated content* (UGC) websites such as YouTube [You08], Dailymotion [Dai08], Metacafe [Met08] as well as many more [Vid08b, Vid08a, Hul08, MyS08]. The most popular of these sites, YouTube, was founded in 2005. Despite only being three years old, it is now the 3rd most popular site on the internet, illustrating how popular these sites are.

	YouTube	Dailymotion	Metacafe
Unique Visitors ($\times 10^6$/month)	70	10	10
Videos Watched ($\times 10^6$/day)	100	25	15
Alexa Site Rank (Feb '08)	3	31	179
Stored Videos (2006)	45 TB	*unknown*	*unknown*
Stored Videos (2007)	357 TB	*unknown*	*unknown*

Table 2.1: Overview of the most popular video sharing websites, adapted from [SSF08, CDL07]

Table 2.1 gives a brief overview of the most commonly used UGC sites' popularity, as well as how many videos are viewed and stored on these services.

These Web 2.0[1] websites allow users to upload their own videos for others to watch freely. Once uploaded, other users can begin tagging [AN07], rating and commenting on each video. Users may also share their favorite videos with their friends, allowing the video to quickly disseminate through a video sharing social network.

Typically the content on the sites is short low quality clips. A study of YouTube by Cheng *et al.* found that 97.8% of all videos were shorter than 10 minutes [CDL07]. This is due to a limit imposed by YouTube, that regular users may only upload videos 10 minutes or shorter. A different study found that the mean viewing length was 4.15 minutes with a median of 3.33 minutes [GALM07]. There is a small group of authorized users who may upload longer videos, which includes content such as documentaries or lectures. These short length videos are not representative of all sites; an analysis of MSN Video found that many of the videos were a lot longer than YouTube, however no exact figures or distributions of lengths were offered [HLR07].

These sites typically use *pseudo-streaming* (or sometimes called *progressive download*) techniques [GCXZ05]. This is where a media file is downloaded using general file downloading techniques, but as it downloads it is also played. As a true streaming protocol is not used, many features are lacking, for example, the rate the server transmission the file will not be synchronised with the playback rate. This may cause the stream to be received faster or slower than required. Additionally controls such as pause or seeking are not easily available.

[1]Web 2.0 is a term to describe a new generation of websites, where the site's value comes from the users who participate.

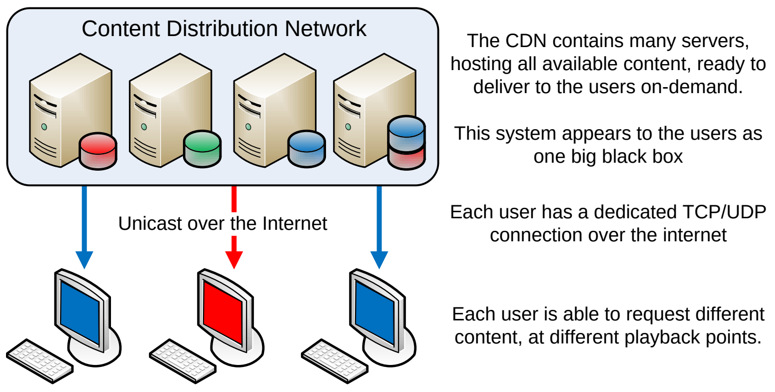

Content Distribution Network
Unicast over the Internet
The CDN contains many servers, hosting all available content, ready to deliver to the users on-demand.
This system appears to the users as one big black box
Each user has a dedicated TCP/UDP connection over the internet
Each user is able to request different content, at different playback points.

Figure 2.1: Distribution of media via a content distribution network. Different coloured arrows and shapes represent the different content.

The most common example of *pseudo-streaming* involves a combination of an Adobe Flash player and HTTP downloading. A web page contains a Flash video player which requests a Flash video file (FLV) from the HTTP server. The Flash file is downloaded just like any other file would be via HTTP. As the file downloads, the Flash player can begin playback.

Flash video was chosen due to its widespread deployment. For example, in 2000, the Flash player was distributed with AOL, Netscape and Internet Explorer browsers. Later, in 2002, the Flash player came pre-installed with Windows XP. This lead to an unverified claim that Flash Player had an install-base of roughly 92% of all internet users [Wik08].

Flash video can be compressed using various encodings. On YouTube, the video is encoded with the Sorenson Spark H.263 codec, with a resolution of 320x240 at 25 frames per second [CDL07]. This creates videos which have a bitrate of around 330 kbps. However, YouTube is at the lower end of quality, compared to other UGC sites. Due to home broadband becoming more ubiquitous, there has been greater demand for higher quality videos. For example, in February 2008, Dailymotion announced that it would begin streaming at the high-definition resolution of 720p (1280x720) [Low08], requiring at least 8-16 Mbps bitrate streams.

The HTTP servers used for streaming are typically hosted by a content distribution network (CDN) as depicted in Figure 2.1. Little is known about how these systems are configured and deployed, however some research has gone into inferring their deployment by taking measurements. In one study, Saxena *et al.* found that YouTube's videos are served from just two main locations in the US; San Mateo (77%) and Mountain View (22%), with the remaining 1% being served by the Limelight's CDN [SSF08]. This suggests most of YouTube's CDN is built and controlled in-house. Saxena *et al.* also found that Metacafe took a different approach and outsourced their needs to the Akamai and L3 Networks CDNs. It is well know that Akamai position their servers into as many ISP's points of presence (POPs) as possible [HWLR08].

Until very recently these UGC sites had limited interactive controls. The Flash player would continue to buffer the video being viewed as fast as possible and store everything which has been received in a temporary file on the computer's hard disk. Users were able to pause, and seek backwards into the stored buffer, but were unable to seek ahead of the buffer. Since late 2007/early 2008, YouTube and others have begun allowing arbitrary seeks anywhere in the media before it is buffered. This has been achieved using a custom client-side Flash player and some server software. How exactly this is achieved is discussed later in Section 3.2, as this technique was developed independently by us before it had been implemented by YouTube.

2.1.2 BBC iPlayer

BBC iPlayer [BBC08] has been leading the way as a new type of desktop application which enables users to watch video-on-demand over the internet, using a normal home broadband connection. BSkyB and Channel 4 have also created similar products to compete with the BBC, named Sky Player [BSk08] and 4oD [408] respectively. These products differ from network PVRs, as they serve a more specific task, and do not require custom hardware.

These products offer a catalogue of old programmes (which are no longer regularly shown on broadcast TV), and access to most of the broadcast programming shown in the last 7 days. Users can select which programme they want, and their client begins to download the video. However, these products do not stream the video; instead they download a single file. This means the user must wait until the full file is downloaded before playback can start.

All three applications are actually based on one companies technology, Kontiki [Inc08b]. This company has created their own technology to provide a content delivery platform, which can securely deliver media from standard servers, assisted by scalable peer-to-peer techniques. Kontiki is closed source software, and as far as we are aware no studies have been conducted to analyse how it works. However, from promotional material, it constructs a simple peer-to-peer network from the users. This network can be configured to limit how the peers are connected, for example, making sure the peers do not connect outside of their own subnet, or autonomous system (AS) boundary.

When a Kontiki client is idle, any spare bandwidth is used to help spread the content within the network. To seed the content into the peer-to-peer network, and to provide additional capacity, a normal network of servers deployed in a CDN are used. This peer-to-peer network does not allow users to publish their own content, as the network is used just as the provider's content delivery platform. To ensure this requirement is met, the network uses strong cryptographic techniques such as asymmetric cryptography [RSA78], to guarantee that media is not tampered with, and also to ensure that new media is not injected into the network without permission.

The files published by the BBC are typically Windows Media (WMV) files, protected with digital rights management (DRM). The DRM stops the files from being shared with others, and expires the files a few days after playback. These files are of good quality, and are roughly 140 MB for a 30 minute show. Since the full WMV files are downloaded before playback can begin, interactive controls are easy to provide. Pausing and seeking to arbitrary points is readily available, and instantaneous.

Due to the Kontiki platform, and the DRM techniques, the iPlayer software only

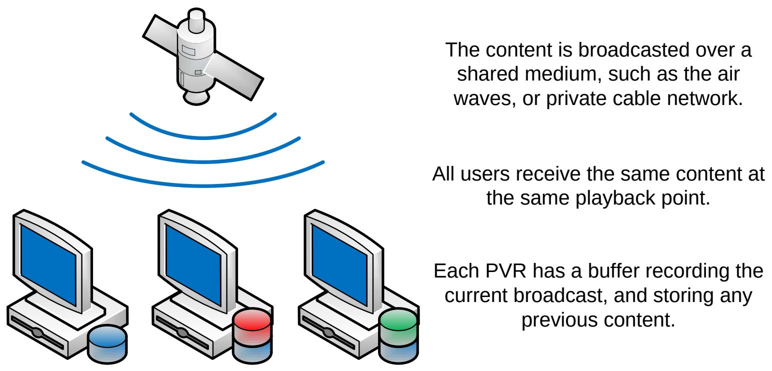

The content is broadcasted over a shared medium, such as the air waves, or private cable network.
All users receive the same content at the same playback point.
Each PVR has a buffer recording the current broadcast, and storing any previous content.

Figure 2.2: Distribution of media in a boardcast personal video recorder network. Different coloured arrows and shapes represent the different content.

runs on Microsoft Windows, and not on other operating systems such as Linux, or Apple's Mac OS. This received many criticisms, as the publicly funded BBC were ignoring a subset of the public that did not use Windows. To counter, this the BBC introduced a new streaming based iPlayer which could be accessed via their website. This uses Adobe Flash technology, similar to that used by many user generated content sites, such as YouTube. By creating their Flash based site, any device which could render Flash video was now able to view the iPlayer's catalogue. This includes desktop computers running various free operating systems, many home gaming consoles, such as the Nintendo Wii [Onl08a] and Sony Playstation 3 [Pur08], and mobile devices such as the Apple iPhone [Onl08b].

2.1.3 Personal Video Recorders

A device that is becoming more commonplace in the living room, is the *time-shifting* set-top box. These devices, sometimes called personal video recorder (PVR) or digital video recorder (DVR), are typically set-top boxes which record broadcast TV. These PVRs allow the user to pause playback while the box continues to record, or rewind within the recording. Additionally, if the user has paused or rewound, they may fast forward to catch up with the live broadcast. It was estimated in 2008 that 36% of the UK uses a PVR [Plc08].

There are many PVRs on the market, the most popular being, Sky+ [Ltd08], TiVo [TT08], ReplayTV [Inc08a] and UltimateTV [Mic08b]. The Sky+ PVR, for example, records broadcast TV received via a satellite dish as seen in Figure 2.2. This PVR has two TV tuners, allowing it to record from two channels simultaneously. As with most of these devices, it contains a large hard disk, able to record within the range of 40 to 80 hours of TV.

The Sky+ box can be scheduled to record future TV programmes or films via the electronic programme guide (EPG). Once recorded, the user is able to play the recorded programmes by selecting them from the EPG. However, this does not give the user a true video-on-demand experience, as they have to wait until the programme airs before being able to watch it. To combat this, Sky+ has recently integrated a *push video-on-demand* system called "Sky Anytime". Sky can instruct the Sky+ box to record popular programming, such as new movies, or sporting events. The programming is sent on hidden broadcast channels, typically during the night when the Sky+ box is not in use. Users can then chose any of these "Anytime" programmes, and play them back instantly from their local hard disks.

The interactive controls are rather limited with these boxes, as they are only able to seek within what is currently buffered. In live TV that means rewinding, but in pre-recorded content (such as "Sky Anytime") they may fast-forward or rewind. To allow rewinding with live TV, the Sky+ box is always recording the current channel, with a buffer of up to two hours. Once the channel changes, this buffer is discarded and a new one begins. This allows a user to pause for no longer than two hours, and rewind the current channel up to a maximum of two hours (as long as the channel is not changed within that time).

Most satellite providers also offer pay-per-view content, such as very new movies, or live one-off sporting events. Events, such as sports, are typically broadcast live on a single encrypted channel, which limits the interactivity to simple pause and rewind. More interesting are the *near video-on-demand* services. These are typically provided for movies, where a single movie will be broadcast on multiple channels using a simple staggered broadcast technique. If staggered, the movie will be broadcasted at fixed time intervals, for example, every 15 minutes. Thus, when a user purchases a movie they will have to wait up to 15 minutes to begin watching.

Near video-on-demand has the ability to allow users to seek forward or backwards in fixed time intervals, for example, 15 minutes at a time. When combined with a PVR, this can be extended to allow more fine grained seeking, if for example, the PVR records two broadcast channels at different positions within one programme. This could allow the PVR to buffer 15 minutes ahead, by using the second channel. Once the first channel has caught up to the buffer, it can begin playback from the buffer, and use the first channel to record 30 minutes ahead. This process can continue until the full movie is buffered to disk. As far as we are aware, no set-top box offers such functionality, mostly due to the added complexity for little gain.

2.1.4 Networked Set-top Boxes (IPTV)

Some set-top boxes, those typically on cable networks, have begun to roll out IPTV services which offer true video-on-demand. In the UK, the main provider is Virgin Media with over 3 million customers using its "On Demand" service [Med08]. British Telecom (BT) have also recently introduced a similar product called "BT Vision" [Tel08]. It is predicted that by 2011, there will be 80 million IPTV users worldwide [OP07].

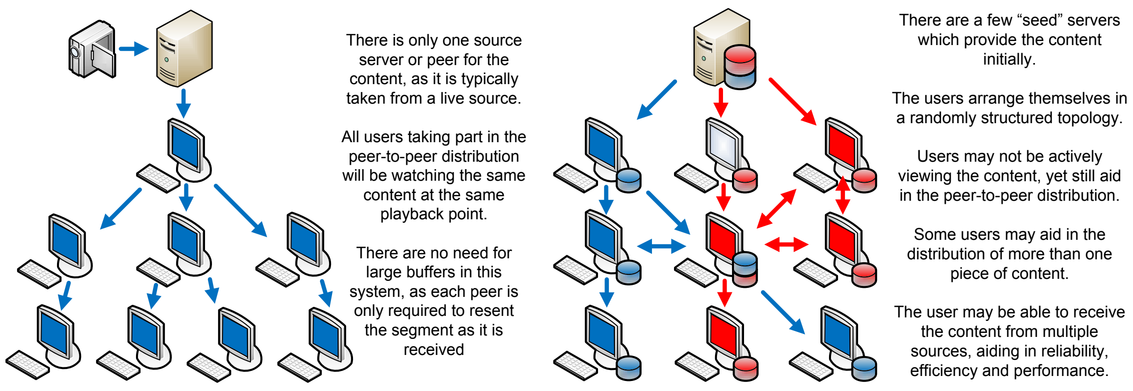
In these systems, a simple set-top box or PVR is connected to either a private network (in the case of Virgin Media), or via the public internet. Content is then streamed directly to the user instantly, on-demand. Virgin Media have been able to offer this service for many years by utilising the existing cable network infrastructure to unicast video from the user's local head-end[2] directly to the end-user. This is not possible in satellite or traditional radio broadcast as both have finite broadcast capacity, whereas cable networks have constantly invested in and improved their networks over the years, adding more and more capacity.

[2]A head-end is a facility run by a cable company to serve customers in the local region

Nevertheless, the on-demand content which is available via these services seems to be of lower than normal broadcast quality, and little is known about the technical details. However, from experiments with these set-top boxes it may be possible to infer how they operate. For example, examining Virgin Media's On Demand service, it is clear that a staggered broadcast system is being used. When seeking through the video, the video jumps in increments of 15 seconds, and when starting a new video it takes up to 15 seconds to begin. This may be because the video is being broadcast in staggered intervals of 15 seconds. It is unclear if this is done so a broadcast technique such as multicast may be used, or if this is to reduce the number of unique channels the server has to transmit.

One feature which Virgin Media does offer, that no other streaming VoD service provides, is the ability to fast-forward or rewind (*e.g.* to view the video at a faster rate either forwards or in reverse). Again, however, this service is limited to just a couple of fast-forward or rewind speeds. Also, when starting or finishing to fast-forward or rewind, the video appears to jump or stutter. This may be because there are dedicated streams broadcasting the video at a higher speed forwards and backwards. Thus, when the set-top box is instructed to fast-forward, it actually joins this different faster stream.

The system being offered by BT is powered by software created by Microsoft named Mediaroom [Mic08a]. This software is also being used by numerous IPTV providers around the world, such as, T-Home (German), Portugal Telecom (Portugal) and AT&T (United States). Most of these providers are using cable or fibre to deliver broadcast quality content to the homes. However, BT have taken a different approach, and are using traditional home ADSL broadband technology. We speculate that this is because of the higher ADSL penetration in the UK, and that cable/fibre networks in the UK are almost exclusively owned and operated by Virgin Media.


There is only one source server or peer for the content, as it is typically taken from a live source.
All users taking part in the peer-to-peer distribution will be watching the same content at the same playback point.
There are no need for large buffers in this system, as each peer is only required to resent the segment as it is received
There are a few "seed" servers which provide the content initially.
The users arrange themselves in a randomly structured topology.
Users may not be actively viewing the content, yet still aid in the peer-to-peer distribution.
Some users may aid in the distribution of more than one piece of content.
The user may be able to receive the content from multiple sources, aiding in reliability, efficiency and performance.

(a) Tree P2P Network　　(b) Mesh P2P Network

Figure 2.3: Distribution of media in a peer-to-peer network. Different coloured arrows and shapes represent the different content.

2.1.5 Peer-to-Peer

Other than the previously mentioned commercial online video-on-demand systems, there are numerous peer-to-peer (P2P) technologies that enable efficient streaming of live and stored media [LGL08, LNZ07]. Traditional P2P has been used to distribute full files [Coh03, eMu08, Cli08], including files such as movies, tv shows, music [Nap08], *etc*. As the full file must be downloaded before playback can begin, this can be considered a primitive form of VoD, similar to that offered by the Kontiki based applications (see Subsection 2.1.2).

After the initial surge of P2P file sharing applications, research began on application level multicast (ALM) [YLE04], sometimes called end-system multicast [CRSZ02]. This is a form of P2P designed to stream content, in a one-to-many fashion, similar to traditional IP multicast. However, streaming is different to video-on-demand, as all the users are typically viewing the same content at the same playback point. Video-on-demand should allow users to view different content at different playback points simultaneously, which makes it much more problematic.

Only recently has research began into the areas of peer-to-peer video-on-demand (P2P-VoD). Using P2P offers many advantages over the traditional client-server approach. Firstly, peer-to-peer can greatly reduce the costs to run the service, as a large

CDN is not required to deliver the content. Additionally, in the mesh-based P2P, those able to request from multiple sources simultaneously have advantages when it comes to interacting with the media. These protocols are designed to allow arbitrary segments to be downloaded. Thus, seeking and pausing can occur relatively easily.

Peer-to-peer does have some disadvantages, such as additional overheads and requiring peers to take part in the delivery. These overheads are from the extra control traffic needed to arrange peers into a structure suitable for media delivery. As peers are the main source of content, they can be less reliable and have less resources than tradition servers. This may result in unpredictable or unreasonable service. Additionally, unless a content protection scheme is used, peers may maliciously alter the content when relaying to other users [DHRS07]. Also, not all peers have sufficient capacity or want to take part in the network.

When describing P2P, there are a few different ways the networks may be arranged and how the data is transmitted.

Tree In tree distribution, the peers arrange themselves in a tree, rooted at the source of the content (as depicted in Figure 2.3a). This is best used for streaming, as there is typically only one source for this kind of content. The content can then be streamed down the tree, and eventually reach every peer. There are many protocols which efficiently arrange peers in this manner [MCH01, BBK02, THD03]. However, it was noted [CDK+03], if nodes are arranged in a tree, the leaf nodes (those at the bottom) do not distribute to others. This is an obvious waste of resources, as f^h peers within the network do not help distribute the content, where f is the node's out-degree and h is the height of the tree. To better utilise the resource, multi-source trees were developed [BAE03, CDK+03]. These multi-source trees also make the distribution more robust to failures, such as, a node parent failing [DHT04]. One problem for all trees, is that they may become very deep, causing a high latency (or lag) for the peers near the bottom.

Mesh To improve the efficiency of tree based schemes, mesh networks create a seemingly randomly connected graph from the peers [KRAV03, LJL+06, MR07, HLR08], as seen in Figure 2.3b. Content may then flow in any direction through this graph, allowing each peer to have many sources, as well as many nodes to share with. This technique is typically used to provide video-on-demand [AGRM06] or file distribution [Coh03], as it offers greater reliability and performance, at the cost of additional overheads and less guarantees of the ordering of received data. Unfortunately, peers within a mesh network will experience a higher rate of churn (peers joining and leaving), as each peer is potentially connected to tens of others.

Typically the content is always divided into segments, of fixed (or sometimes variable) size. This allows peers to request individual chunks of the media, as well as to more efficiently inform others which segments they has. The segments are normally either pushed or pulled by the peers though the peer-to-peer network.

Push Push distribution is typically used for live streaming in combination with tree based distribution. In live streaming each peer will require the segments by a similar deadline, and each peer typically only has one parent, the segments can be pushed to the peer, without request. This greatly reduces the overheads of knowing which peer needs which segment, *etc*. Push can also be used in multi-source situations by clever partition tricks, for example, in a two-source situation, a peer can receive odd segments from one peer, and even from the other. However, this gets increasingly complex when there are multiple source peers or the network is in a constant state of churn (as is common in mesh networks).

Pull If it is not obvious which peer needs which segment, then pull distribution is better. Each segment must be explicitly requested by the peer before it is sent. This adds additional overheads, but allows the receiving peer to make decisions on where to receive from. Pull is popular in mesh networks, as it simplifies the distribution of content. To allow pull to work, each peer must occasionally share a list of currently buffered segments with their neighbours. This list is typically shared in the form of a bit-map, assigning a one or zero to each segment, indicating whether it is buffered or not.

This rest of this section discusses the main peer-to-peer systems in both the streaming, and VoD domains.

2.1.5.1 Streaming

Streaming, is typically used for live events, or broadcasting of traditional style television channels. As such all the users will be viewing the content at the same playback point, as opposed to video-on-demand, which allows users to view different playback points simultaneously.

There are many commercial peer-to-peer streaming products available, mostly from Chinese companies. These include PPLive [PPL08], CoolStreaming [ZLLY05], Zattoo [Zat08], TVAnts [TVA08], PPStream [PPS08] and SOPCast [SOP08]. This software has been very popular in China [FM05], and is starting to become more popular in the US/UK.

CoolStreaming (or more formally known as DONet) was one of the most popular services, when it was in operation. Information about how the system works was made publicly available, and a couple of papers were published on the topic [ZLLY05, XLKZ07]. However, in 2005 the service stopped broadcasting, less than a year after it first began, due to copyright issues.

The CoolStreaming technology is based on a pull mesh-based streaming technique. When joining the system, a newly connecting peer would obtain a list of existing peers from a central repository. This list would be used to bootstrap the newly connecting peer into the network. Afterwards, a gossip protocol is used to find additional peers. Segments of the media are there pulled from neighbouring peers, who frequently advertise their segment lists.

One of the novel features of CoolStreaming is the scheduling algorithm which decides which peer is used for a segment when there are multiple peers to chose from.

The problem of deciding the most efficient way for each peer to allocate it's resources is an NP-hard problem, akin to parallel machine scheduling [CLRS01]. Therefore, CoolStreaming uses a simple heuristic to decide the allocation of resources. The algorithm uses a combination of how rare the segment is, how much free bandwidth the remote peer has, and how urgently the segment is needed.

PPLive [PPL08] is more popular than CoolStreaming, as it has a total of 2.2 million users and 500 different streams [HFC+08]. A keynote presentation by Huang, a PPLive Software Architect, demonstrated how scalable P2P streaming can be. In the second quarter of 2007, PPLive supported 1,480,000 simultaneous users viewing the same live sporting event, being served by just one 10Mbit/s server [Hua07].

Even though PPLive is a closed-system it has been a hot-topic for researchers to study [HFC+08, SF07, VGLN07, KS08, CCL08]. Silverston and Fourmaux captured traces from PPLive, and determined it uses a mesh-based pull approach, similar to CoolStreaming [SF07]. Vu *et al.* noted that PPLive tries to keep its neighbour peer list around 30 to 45, independent of the number of peers currently taking part in the stream. By keeping the neighbour list around a constant size, this allows the system to scale far more efficiently [VGLN07].

Vu *et al.* also calculated the *clustering coefficient* [WS98] of this network. This is a measure of how randomly the peers are connected to each other. They found streams with few peers (< 500 peers) had a high degree of randomness, however, as the stream size increased, many clusters of peers began to form. They did not speculated as to whether the clusters were based on some metric of "closeness", *i.e.* network or geographical locality.

The remaining studies which look at PPLive have looked at simple metrics such as packet size [KS08], signal overhead [SF07], stream popularity, and chunk availability [HFC+08]. These do not give much insight into how PPLive operates. However, one thing is clear, PPLive is a large peer-to-peer application which has tremendous scaling abilities. This is only let down by the fact that it is a pure streaming application, and does not offer interactivity features beyond pause and resume. Nevertheless, PPLive will encourage development of future projects which take advantage of this form of streaming peer-to-peer, hopefully with addition VoD features.

2.1.5.2 Video-on-demand

Peer-to-peer video-on-demand (P2P-VoD) typically uses a combinations of P2P file sharing and streaming techniques. Users will contribute local disk space, as well as bandwidth, to allow other users to stream directly from them. Typically the local disks will store multiple videos which have been previously watched, and perhaps a few which have not if the network deemed their replication necessary. Many of the pure streaming techniques can be applied, or slightly altered, to work for video-on-demand. However, this is easiest with the pull based systems, which can easily cope with peers being at different playback points.

There are a number of commercial systems, such as, Vuze [Vuz08], Joost [Joo08] and many others [Gri08, PFS08, PPS08, UUS08]. Again, all of these systems use proprietary techniques, and as such the only information about them is inferred, or discovered through measurements.

Vuze, for example, offers a catalogue of thousands of videos, mostly uploaded by users, but some from professional studios. To download the videos Vuze uses a sliding window BitTorrent [Coh03] technique [VIF06a, SP07b]. To begin viewing a video, Vuze must connect to a tracker. The tracker is a centralised server or possibly decentralised in some modern BitTorrent implementation [Roo06]. The tracker maintains a list of all peers who are in the process of downloading, or have finished and now just sharing. The Vuze client uses this list to form a single P2P network for each video.

Normally, BitTorrent connects to as many peers in the P2P network as possible and begins downloading. Multiple downloads occur in parallel, each requesting a different random segment of the full file[3]. The random order helps ensure that the

[3]BitTorrent does not always download segments in a random order, as there are multiple improvements to increase the efficiency of the ordering [MV05]

file is spread as quickly as possible throughout the network [BHP05]. So that Vuze can display the video to the user as it downloads, it opts to download the segments of the file in a semi-sequential order. A sliding window is created ahead of the playback point, and only segments within this window are downloaded. As playback continues, the window moves along.

In theory, Vuze could support many interactive controls, however, it only supports pause and resume. Pause is a simple operation as playback from the buffer can stop while not affecting the normal download. Seeking is not possible as time offsets cannot be easily mapped to file segments. This feature could be added if metadata provided a map of keyframe times to an offset within the file. Then on a seek request the sliding window can be moved to the new seek location, and download/playback resume.

There are numerous problems with Vuze, for example, the protocol is a very simple modification to BitTorrent, which is not custom-made for this task. This causes the start-up times to be long, and limits the interactive features. Also, because the peer-to-peer networks are only made up of peers who have previously downloaded, or are downloading, the video, it is possible for the video to not be fully available. A more suitable situation would be to either backup the videos on dedicated content servers, or ensure the videos are replicated on nodes with spare capacity, therefore better utilising the network.

Joost [Joo08], takes a different approach to Vuze, by designing a new P2P-VoD protocol from the ground up. Joost, was created by Niklas Zennström and Janus Friis, the two entrepreneurs responsible for Skype [Sky08] and Kazaa [Net02]. Joost has a large catalogue of content, which is provided exclusively by professional studios. Because of Skype's and Kazaa's fame, Joost has been able to secure deals with many large studios, such as FOX networks, Viacom (which includes MTV and Paramount Pictures), and Warner Music. This has allowed Joost to have high quality content, such as feature films and TV episodes.

	HTTP Flash	PVR	Networked PVR	Peer-to-Peer Push	Peer-to-Peer Pull
Overheads	Minimal	Minimal	Minimal	Small	Small-Moderate
True VoD	Yes	No, but offers Near-VoD	Yes	No	Yes
Pause	Yes	Yes	Yes	Yes	Yes
Seeking	Yes, with Seeking Hack	Yes, within Buffered Content	Yes	No	Yes
Fast-forward & Rewind	No	Yes	Yes, with limitation outside of buffer	No	No
User Costs	Internet Access	Set-top Box & Subscription		Internet Access	
Provider Costs	Large CDN / Server	Broadcast Infrastructure	Broadcast & Network Infrastructure	Small Seed Server	
Quality	Low–Medium	High	High	Medium	Medium
Pros	Simple Protocol, Small Seek Latencies	Easy for users	Private network can be optimised for the VoD task	Cheap, Lots of Content	
Cons	Costly and Limited Interactivity	Limited VoD & Interactivity	Costly	Complex Protocols, Long Start-up and Seek Latencies	

Table 2.2: Summary of features available with each video-on-demand system

Little is know about Joost, however it is reported that it is a mesh based peer-to-peer network backed by servers deployed in a CDN. From one study, it appears that Joost uses UDP packets, to transmit content in MPEG-4/AVC with error-resilience coding [KS08]. It is speculated that Joost will use peers with spare capacity to help distribute content which is popular. This would help to maximise the delivery efficiency.

2.1.6 Summary

The previous sections have outlined popular video-on-demand systems which are currently deployed and in use. Their features have been explained, as well as the pros and cons of using them. Here, we will summarise the previous sections, and aim to discuss the systems compared to each other. To recap, Table 2.2 lists the main categories of systems, and which features they support.

The HTTP Flash-based systems are typically backed by a content distribution network (CDN), but can in small cases be simple client-server systems. By using a simple *pseudo-streaming* technique, the flash-based web-site is able to provide videos on-demand to a vast audience of users via the internet with minimal overheads and costs to the user. However, the back-end system, will no doubt involve terabytes of replicated data, spread across hundreds if not thousands of servers, typically deployed throughout the world. The cost of running such an infrastructure is not cheap, for example, in 2006 it was estimated that YouTube pays $1 million a month just for bandwidth costs [Fro06]. As such, only the well funded content providers can afford to provide this service.

Content distribution networks used by Flash-based systems can provide many of the modern interactivity features requested by users, albeit with additional overheads and complications for the servers. The one downside when interacting with these systems is the seek-delay. This is normally quite small (less than 2 seconds), and no longer than a couple of round trip delays, and the time it takes to fetch an initial buffer. This can be reduced by making sure the content servers are near to the end users.

Personal video recorders (PVRs) are perhaps the simplest system for users, as they integrate with users' existing home entertainment systems. If the content provider is already broadcasting the content, then the cost for deploying the PVRs is just the price of the box. However, a simple broadcast-only PVR does not allow for true VoD, only being able to watch pre-recorded content. To add true VoD, dedicated networks are typically used, which greatly increase the cost for the provider. Regardless, PVRs are able to record content at broadcast quality, which is much higher than that typically found on the internet.

Finally, the peer-to-peer model of distribution offers the cheapest way to deliver content, and if it is not streamed (and instead downloaded) the highest quality of content. This allows independent movie studios, or amateurs to easily release their work in high-def quality formats. Being cheap comes at the cost of requiring users to aid in the distribution, which typically involves high signalling overheads, for example, to coordinate all the peers in a distributed manner. Aiding in distribution is unappealing to many, as they either have to pay for their bandwidth, or are unwilling to share their resources if they are required to pay for the content or service.

Peer-to-peer also offers numerous other challenges, which can vastly affect the performance of the system. Unlike with CDNs and PVRs, the relative simplicity of the protocols allows them to have a high level of service and reliability. However, in peer-to-peer, your level of service depends on other users, who join and leave the network as they choose. Also, if a user decides to be malicious, they may disrupt the network, inject illegal content, or tamper with the existing content.

The added complexity of P2P does allow for higher levels of interactivity. For example, in pull based P2P, the content may be fetched out of order. It is therefore trivial for the protocol to fetch new seek points, or to even pre-fetch ahead to areas of interest. Features like this reduce seek delay, but this improvement may be negated because of the high overheads and unreliable performance of other peers. There is certainly room for much improvement and innovation to solve the numerous challenges found within P2P.

2.2 Characterisations of User Behaviour

To design systems that support the delivery of multimedia over the internet, it is crucial to understand how users will interact with the media. These interactions impact multiple functions, such as, admission strategies, buffer management and delivery techniques.

In a content distribution networks (CDNs) context, this could effect how proxy servers operate, and how the location of replicated media, and which delivery mechanisms are used. If, for example, only a small subset of media from a large catalogue is popular then more resources should be dedicated to those popular files. Content could also replicate in advance, if it was possible to anticipate demand. This is all possible by understanding how the content is consumed by the user.

One area which has not been closely examined, is when there are high levels of interaction, such as those when a user wishes to view just the highlights of the content, or is searching for a specific clip. Understanding highly interactive characteristics can aid in the designs of many novel features. This may include the ability to pre-fetch areas of high popularity, or place bookmarks at key points.

The behaviour of users has previously been studied in a few different domains, including static web content, video-on-demand, and live streaming. Each have different properties, causing the observed user behaviour to differ for each domain.

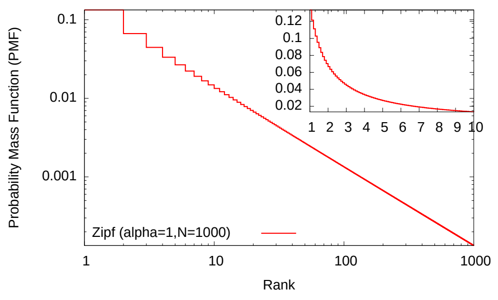
Video-on-demand (VoD) domain consists of applications in which videos from a stored catalogue can be fetched and viewed at the user's discretion. These applications typically allows the user to control the playback of the content, for example, allowing users to pause, fast-forward, or rewind. Live streaming is akin to TV broadcasting; all users viewing a particular stream do so at same playback point, thus limiting their control over playback.

Veloso *et al.* describe VoD as *user driven*, meaning that the user decides which media is viewed and when. However, live streaming is *object driven*; the user's access is influenced by show/event time, and the various activities within the live media [VAM+02].

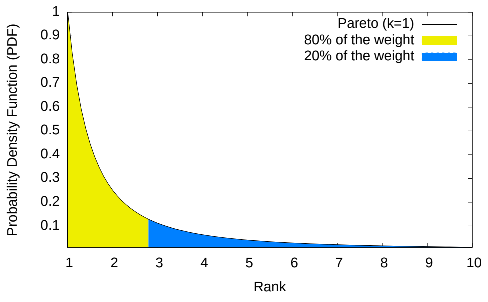
2.2.1 Video-on-Demand

The characterisation of Video-on-demand (VoD) is important to this thesis, as the main experiments involved VoD. As such it is important to have a understand of existing VoD characterisations, to contrast with the new results found within this thesis. Additionally VoD has become increasingly popular over the internet, and is thus introducing new challenges which need solving. Already, 11% of people within the UK supplement or replace their broadcast TV viewing with online video services [Plc08].

In a 2001 study, the streaming habits of users on the University of Washington campus were recorded. It was found that 85% of all videos viewed were from stored content [CWVL01]. This percentage is thought to have increased as numerous video websites have become very popular, with YouTube [You08], for example, receiving 70



(**a**) Zipf distribution: The i^{th} rank occurs $1/i^{\alpha}$ of the time. This gives the appearance of a straight line on a log-log plot



(**b**) Pareto distribution: This follows the 80-20 rule, where 80% of the function's weight is within the top 20%

Figure 2.4: Probability density functions for Zipf and Pareto distributions

million visitors a month [SSF08].

Multiple studies have suggested that the majority of online VoD content is relatively short [LCKN05], with 93% of content having a duration of less than 10 minutes [CWVL01], and a median of 3.2-3.9 minutes [SSF08]. The length of video content is expected to increase as VoD becomes more popular driven by services such as BBC iPlayer [BBC08] offering TV shows and feature films.

2.2.1.1 Popularity

The popularity of content within a VoD system play an important role in deciding how it is cached and replicated. Popularity of web objects typically follow a Zipf-like distribution. Within Zipf distributions [Zip49], the popularity of a object is proportional to its rank, *i.e.* the i^{th} most popular object receives $1/i^{\alpha}$ of requests, as seen in Figure 2.4a. This implies the majority of content is unpopular, and a few items are extremely popular, making up the weight of the distribution. This is be illustrated by observations made by Chesire *et al.* Out of 23,738 video objects, 78% of which were only accessed once, 21% accessed two to nine times, and the remaining 1% accessed ten or more times, with the 12 most popular objects being accessed more than 100 times each [CWVL01].

Video-on-demand popularity was first suggested to follow a Zipf distribution by Dan *et al.* in 1994 [DSS94]. This was again observed by Wolf *et al.* in 1997, however with varying degrees of skew each week [GBW97]. In early 2000, Acharya *et al.* could not fit the popularity of their education videos to a Zipfian distribution [ASP00]. Instead they noticed requests to their objects were more biased towards the popular titles than expected within a Zipf distribution. This bias can be explained by Cherkasova and Gupta's analysis of an enterprise media workload [CG02]. They observed that over long timescales the bias towards the most popular items increased. For example, object popularity fitted a Zipf distribution with $1.3 \leq \alpha \leq 1.6$ over 1-month periods, over a 6-month period with $\alpha = 1.6$, and eventually Zipf did not fit well over a year.

The fact that Zipf did not fit well over a year time scale is often overlooked and the reasons for this are typically misunderstood. Zipfian models are useful for static distribution, those which do not have a temporal component. For example, Zipf was first developed by George Kingsley Zipf when studying the frequency of words appearing in a corpus of natural language. The corpus would not change over time, *i.e.* words would not be added or removed. This differs from popularity of videos over a timescale as it is common for new videos to be added and removed over time, and for the popularity of the videos to change over time. Instead the Zipf models should be used to model the popularity over shorter periods, such as daily or weekly, or be used to blindly model the rank of objects on daily bases. For example, objects on a daily bases may follow a Zipf distribution, but instead of noting the popularity of each object, note the popularity of each rank. This will then more accurately model the expected popularity on any given day, and is more useful for the design of caching systems.

Whilst analysing Kazaa's [Net02] peer-to-peer traffic, Gummadi *et al.* proposed a new distribution that fitted the popularity of objects better than Zipf [GDS$^+$03]. Individual Kazaa users rarely requested the same object twice, unlike in web traffic where the same object may be requested multiple times by an individual. This lead to the "fetch-at-most-once" model, which fitted better to the workloads discussed by Cherkasova and Gupta [CG02], as well as ranking data collected from video store rentals [Com00], and box office sales [Dat03]. This model says that users fetch objects following a Zipf distribution, but must not request the same object twice. If the user chooses a previously fetched object, then a new object is picked again from the Zipf distribution.

Yu *et al.* contradicted the fetch-at-most-once model when analysing a 219 day trace collected from a VoD system deployed by China Telecom used by 150,000 people [YZZZ06]. Their object popularity fitted best to a Zipf distribution, except for a long heavy tail. They speculate that their results do not fit the fetch-at-most-once model because users were unable to save the viewed media, thus if they wished to watch a video again, they had no choice but to re-fetch it.

Which model is best seems to depend on how the videos are accessed, what genre of video they are, and other currently unknown factors. However Cha *et al.* tried to explain the shape of the distributions through simulation [CKR+07]. A fetch-at-most-once model was simulated with a Zipf distribution of $\alpha = 1.0$. Parameters such as the number of users, the number of requests per user, and the number of objects were varied. They found that the effects of fetch-at-most-once are barely noticeable when there are few requests, this intuitively makes sense as there is less chance of selecting the same object twice. The number of users did not seem to impact the shape of the distribution at all. When the number of objects increased, the effects of fetch-at-most-once were also reduced, again, because the chance of selecting the same object is decreased.

Both Zipf and fetch-at-most-once are categorised as power-law distributions. However there are other factors which govern the power-law nature of these distributions. In most of the analysed models, the two ends of the distributions have been truncated or extended. It is suggested by Chris Anderson that "Latent demand for products ... is suppressed by bottlenecks in the system" [And06]. Take, for example, the popularity of movies in cinemas. Most cinemas show the most popular movies, however there are few cinemas screening niche content. This causes the popularity distributions of the movies to have a truncated tail. This is refereed to as a "distribution bottleneck", where due to lack of distribution, the tail is truncated. The opposite can be true, where, for example, there is ample supply of niche content, but it is hard to find this niche content, thus a "information bottleneck" exists [CKR+07].

2.2.1.2 Popularity Over Time

Over time the popularity of videos will change, greatly impacting the decision on what to cache or replicate. It is essential to consider how frequently replication updates are carried out. Too often and video may needlessly be moved, but too infrequent and the servers may not be prepared for demand of a newly popular video. As such the rate of change in user interest can aid in the design of a VoD system.

Paneda *et al.* noted the popularity of videos on a popular Spanish news website would change daily as new content was added. Typically the most popular items for each day were added to the site on that day. However after the first day the content could be grouped into four categories; *short life*, *long life*, *up and down* and *seasonal*. Short life content would reach its peak popularity on the first day, and after the first day the number of accesses would decrease suddenly. Long life content also peaks on the first day, however its popularity decreases slowly over the the following days/weeks. Up and down content, will build up popularity for a few days, and then decrease in a similar way to long life. Finally, seasonal content would have peaks in popularity every few weeks or months [PGM+06].

In a study of a enterprise media server, Cherkasova and Gupta observed that on any given month most of the bytes transferred were for new content. They found that ~50% of requests to content were made in the their first week, with and addition 20% to 30% being made in the following four weeks [CG04]. They did not discuss if they found seasonal or *up and down* style content. However these patterns may be directly related to the genre or appeal of the content.

Yu *et al.* took a different approach to monitoring popularity over time in their large scale VoD system [YZZZ06]. They looked at the rate of churn in the top-10, top-100, and top-200 most popularity movie charts over different time scale. They began by looking how newly inserted content affected these top charts on a hourly timescale. The time content was added to the system, correlated with a high change in user interest. This encourages that newly added content be replicated early to avoid insufficient availability.

It was also noted, that over a small time scale, *i.e* hours, the top-10 list had a small amount of churn averaging around 25% change per hour, whereas the top-100 and top-200 experienced 45% change per hour. Over longer timescales such as days and week, it was found that the top-10 list rarely remained stable, whereas the top-100 changed only 15% each day.

The main observations from Yu *et al.* suggest that popularity changes on different timescales, with the top-10 being stable for a day, and the top-100 being stable over longer periods. It is suggested that a two level caching model be used, with a small adaptive cache for the top-10, and larger more constant cache for the top-100 [PGM$^+$06].

Although popularity of objects change over time, it appears these changes are very specific to the viewing population and genres of the objects. In the current research no single model has been found which accurately explain the observed behaviours, however, it is clear that this is an important metric for cache design.

2.2.1.3 Recommendations

Some systems display a top-10 list of the most popular videos that month, or a list of recommendations, such as new releases, or videos a user may find of interest. All of these lists can greatly influence what a user views. However, the importance of recommendations has not been studied, much, it has been highlighted as a great source of revenue for companies such as Amazon [And06].

Yu *et al.* have studied this phenomenon more analytically when they observed it in their VoD system. They found one video stayed in the top-15 most popular film list for a significant amount of time, but once a administrator manually removed the film from the list, its popularity quickly declined [YZZZ06], and never recovered.

The fact that users appear influenced by these lists, can play in the favour of a video-on-demand system. For example, if it is know ahead of time that a film will appear in the list, the system can take necessary steps to ensure the content is well replicated, in advance of the demand.

2.2.1.4 Session duration

The session duration can be defined in two ways; firstly the duration a user spends using a Video-on-Demand application, and secondly the duration the user spends watching a particular video. Each definition is applicable in different contexts. For example, knowing how long a user views a single video can aid in caching decisions, whereas knowing how long a user uses an application can aid in the design of the application. This section is only interested in how long a user views a particular video for.

From early studies on VoD it has been observed that session duration is quite short. For example, in 2001, whilst studying streaming traffic on a large university campus, Chesire *et al.* showed that 85% of all sessions lasted less than 5 minutes with a median session duration of 2.2 minutes. This is compared to a mean advertised media length of between 2.5 and 4.5 minutes. Long lived sessions (those longer than one hour) accounted for only 3% of all client sessions [CWVL01].

Almeida *et al.* found similar results when analysing logs from an educational media server. A significant proportion of requests were less than 3 minutes in duration. These sessions were very short when compared to the length of the media, which had a median length of 60 minutes [AKEV01].

It was suggested by Guo *et al.* that the short duration was due to the long wait times, and low patience of users [GCXZ05]. Yu *et al.* found that short durations within their traces were due to users sampling the media by "scanning" through them. An inverse correlation was also found between the session lengths and the popularity of the media. Less popular videos actually had longer session times [YZZZ06].

As noted by Guo *et al.*, 20% of network bandwidth was wasted buffering video which was never watched due to the user aborting the stream [GCXZ05]. These short session durations encourage the design of agile systems which can quickly display the media before the user becomes impatient, as well as techniques such as prefix caching [SRT99], which prioritise caching of the first frames of the media for quick delivery.

2.2.1.5 Interactivity

The playback of media is not always passive; certain systems allow for interactive control over the playback. For example, the user may be able to pause, fast-forward or rewind, as well as seek to arbitrary points within the content. Offering interactive features can be challenging. For example, most multicast delivery techniques require all clients to be at a similar playback point within the stream. However, if a client seeks arbitrarily, they are no longer at the same playback point, and thus must join another multicast group, or start to receive the content via a different delivery method. As such, an understanding of how users interact with the content can be invaluable for good delivery.

There have been few studies on how users interact with media. This may be due to the relatively few systems which have interactive features. However, Huang *et al.* found then when interactive controls were available, for videos clips shorter than 30 minutes, only 20% of all sessions showed interaction from users [HLR07]. Unsurprisingly, the longer the video, the more interactivity is observed. For example, with videos less than a hour in length 40% of session exhibited interactivity. This trend is consistent with the results within this thesis, and other research [CCB$^+$04]. However, this thesis presents results with far higher percentage of sessions with interactivity. This may be due, in part, to the novel interface presented to the users which encouraged interactivity.

Costa *et al.* found that when users do interact, that in both education and entertainment the most common action is pause [CCB+04]. This was confirmed by Vilas *et al.* who noted that 7% of session for short videos (those less than 5 minutes in length) and 10% of longer videos session had at least one pause. It was also shown that the pause duration could be modelled with a Weibull distribution, with means of 55 seconds and 95 seconds, for short and long videos respectively [VPG+05].

Another common action is seeking forwards or backwards. For short videos, the percentage of forward and backwards seeks appeared to be roughly the same. However as video length increases, there are more forward seeks, indicating users wished to skip ahead [AKEV01]. Seeking forward also surprised Padhye and Kurose when studying an education server which provided lectures. They assumed students would regularly seek backwards to go over a section again, but found that seeking forward was seven times more popular than seeking backwards [PK98].

Vilas *et al.* modelled the number of seeks per session, and follow it matched a Zipf distribution, with α values between 3.73 and 5.8 [VPG+05]. This implies that most users never sought backwards or forwards, however when users did, they did so numerous times.

The distance sought was also studied by Padhye and Kurose, who observed on their education server a very large average distance. For forward seeks this was approximately 35 minutes, and backward was 34 minutes, for media around 70 minutes in length. However, one third of these seeks were for less than 3 minutes [PK99].

2.2.1.6 Segment Popularity

When users seek, not all segments of the video may be viewed equally. This could lead to some segments being very popular, whereas others unpopular. This situation can also be caused if users do not watch for the full duration of the video. This all has implications on how media, or segments of the media should be cached.

Almeida *et al.* divided education media into ten-second segments. For the most popular video, all segments were accessed roughly equally, however for the less popular video, the earlier segments within the media were accessed more [AKEV01]. This indicates only the beginning of the content was viewed. Costa *et al.* also observed this result with newer education content, as well as entertainment content [CCB$^+$04].

When Huang *et al.* analysed the traces from a large entertainment Video-on-Demand site, they found users would regularly quit before the end of the video. For short videos, most users watched for the full duration, however as the video length increased users were more likely to stop early. For example, with videos less than 30 minutes in length, only 18% would watch for the full duration, with 22% watching for 60% of the duration [HLR07].

These later results are consistent with the findings in this thesis, however, we noted areas of high interest dubbed *hotspots*. The previous work found only minor differences in segment popularity, whereas we found segments with orders of magnitude different popularity. This result is speculated to be because of the higher levels of interactivity found within this thesis' traces.

2.2.2 Live Streaming

The characterised workloads of live streaming are likely to be different to those of video-on-demand for a couple of reasons. Firstly, users are mostly passive in live streaming, fairly limited by how they can interact with the media. In some cases, pause is available, but seeking is typically not. The only real interaction available is the choice of when to join or leave the stream. Also, as all users are viewing the same content, the force of the crowd may be stronger, for example, all user leaving simultaneously as a live programme ends.

Secondly, in a 2004 study of a large CDN, it was found that only 7% of streams were video, accounting for only 1% of all requests [SMZ04]. The remaining streams were audio only, for example, radio stations. This section highlights the differences between VoD and live streaming workloads, while explaining any features specific to live streaming.

2.2.2.1 Time-of-day

Non-stop streams have a diurnal access pattern, peaking at the same time each day. This kind of diurnal access patterns has not be observed with Video-on-Demand. It is speculated by Veloso *et al.* that diurnal access patterns are smoothed out because the clients have control over when they access the media and by clients accessing from multiple time zones [VAMJ$^+$06]. The non-stop stream is also influence by time zones, but less so, for example, Sripanidkulchai *et al.* observed for a single radio station's stream, several similar periodic patterns were present, but shifted by the time zone of the clients [SMZ04].

Streams with a short durations did not follow the same daily pattern. However, nearly all short stream began with a *flash crowd*. A flash crowd is a sudden surge of users all wishing the view the same content. This typically overwhelms servers, and results in an accidental denial of service. These types of events are certainly *user driven*, with users specifically joining the stream for an event. Almost 50% of non-stop streams also exhibited a flash crowd event every few days. This, for example, could be the result of many users joining a radio stream to listen to a popular programme [SMZ04]. This behaviour is rarely seen in VoD systems, as the requests are *object driven*.

2.2.2.2 Session duration

Session durations for live streaming follow that of video-on-demand. Most sessions are short, with a a few long-lived sessions. Vandermerwe *et al.* found the distribution of session duration to be long-tailed, with 69% of sessions being less than 2 minutes in length, 88% less than 10 minutes, and the top 8% being longer than 20 minutes [VdMSK02]. These results were similar to Chesire *et al.* who found a similar long-tailed distribution with their top 3% of the population being more than a hour in length. However, these 3% of long-lived sessions accounted for about half of all bandwidth consumed [CWVL01]. This demonstrates how the distribution is not just long-tailed, but heavy-tailed.

Sripanidkulchai *et al.* compared the session durations of live streaming content, repeating streaming content and video-on-demand content. The repeating streaming content consisted of non-stop streams which broadcasted the same programme over and over, for example, every 30 minutes. The session duration for both repeating streaming content and the VoD content exhibited similar "truncated" Pareto distributions. The majority of sessions with this distribution are short, with the few long-lived sessions being no longer than the content's length. However, the session durations for live streaming content, fitted a Pareto distribution with a heavy tail. This extended long tail is caused by the user's actions, rather than being truncated arbitrarily by the content's length [SMZ04].

2.2.2.3 Interactivity

As far as we are aware there have been no studies on the characterisation of interactivity with live streaming. This will of course be due to the lack of interactive controls. For example, it is impossible to seek forward in a live stream, and only possible to seek backwards if the stream has been stored by the server or client.

The storing of live streams by the client is becoming increasingly popular with the use of *time-shifting* devices such as digital video recorders (DVRs) or personal video recorders (PVRs). These are typically set-top boxes which buffer broadcast TV. The user is then able to pause while the box continues to buffer, or rewind within the buffer. Additionally, the user may fast-forward to catch up with the stream if they have paused or rewound. PVRs are discussed in more detail in Subsection 2.1.3.

As far as this author is aware, there have been no research examining how users interact with PVRs. However, it is commonly reported that these devices are used to fast-forward through advertising when watching recorded programming [BP05]. When users skip adverts, these advertising segments become less popular than the rest. This has implications for how these segments would be cached or replicated.

2.2.3 Implications

This section has highlighted the main characterisation models for both video-on-demand and live streaming. Understanding how media is consumed has many practical uses, such as designing, evaluating, planning and managing systems. This is especially true when dealing with highly interactivity workloads, as these typically cause excessive load for servers.

It is clear that starting with a metric as simple as content popularity can greatly aid in replication and caching strategies. As popularity is typically modelled by a power-law distribution, systems will benefit from caching the most popular content. However, as distribution bottlenecks are reduced, and users find it easier to access their niche content, it might be useful to implement multi-level caching hierarchies, which can use different policies based on the the ranking of the content. For example, one caching policy can be used for most popular content, whilst another can be used for the more niche content. As such, the niche caching policy may only store the content in local caches for a short, whereas the very popular content is kept available for a longer period of time, on a more global scale.

Knowing what is popular, and caching it, is a reactive method, however, it is sometimes useful to be proactive. This may be possible if the content is listed in top-x charts, such as the top-10 voted movies. The popularity of an object ranked 10th , is far higher than that at position 11, when only the top-10 chart is available to the user. The effect these charts have on the user's viewing habits can easily be exploited by the video-on-demand system. As soon as the chart is available, the system can begin pro-actively replicating the content, perhaps near to the target demographic.

It is also clear that certain content has a seasonal or recurring monthly appeal, such as Christmas-themed videos. Again, these can be pro-actively replicated ahead of demand, at the cost of potentially wasting resources. Other content is perhaps only popular for a very short time, such as daily news reports. Understanding the appeal of the media can greatly help choose on the appropriate content management techniques.

Where interactivity is concerned, it is already clear that users can be impatient and either scan through the content, or prematurely stop playback. Therefore, to maximise caching efficiently, the first segments of media can be stored in preference to the later segments. Techniques such as prefix caching [SRT99, HNG^{+}99] can also aid in deciding which segments are the most useful to cache and replicate.

Even though users do interact with the content somewhat, high levels of interactivity have not previously been reported. However, the results presented later in this thesis (see Chapter 4) show much greater levels of interactivity. Different design decisions must be made under these workloads, which will be discussed later in the thesis.

Chapter 3

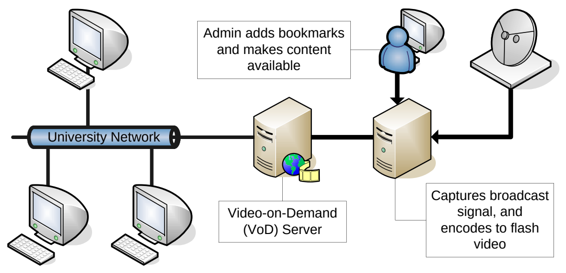
Interactive Video-on-Demand System

There has been lots of work on characterising user behaviour when viewing video-on-demand (VoD) and streaming media (as discussed in Section 2.2). However, there has been little analysis when a highly interactive system is used, for example, a system where users regularly pause and resume playback and actively seek around the media. This would generate results which would be a complete departure from the classic start-to-end model.

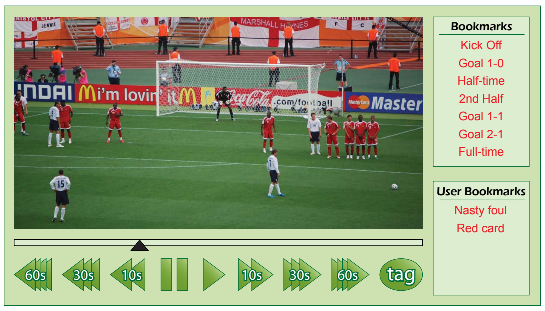
To obtain traces from a highly interactive workload, we set up and designed a video-on-demand system. This system was designed to provide powerful, yet simple interactivity controls, which would hopefully encourage more interaction between the users and the system. Once deployed, the system would be used to record traces of real user behaviour, and to be used as a test environment for future experiments. To be useful, the system had to meet three criteria:

Wide user base So that we could maximise the number of participants, the system needed to be designed in a way which was non-invasive, and simple for the users to use. This ruled out installing any special software on the users machines.

Encourage Interactivity To ensure that the system generates a highly interactive


Admin adds bookmarks and makes content available
University Network
Video-on-Demand (VoD) Server
Captures broadcast signal, and encodes to flash video

(a) System setup



(b) Player interface

Figure 3.1: Diagrams of our video-on-demand system

workload, the layout of the user interface should make it easy to pause, resume, seek, *etc.* This should be supported by the system, which should offer these features with low-latency, as to not to discourage their use.

The content chosen for the system is also important. It should somehow encourage the use of interactive controls, for example, feature films will always be viewed start-to-finish, but genres such as sports may encourage users to view just the highlights.

Simple and Cheap Finally, to ensure this system could be deployed, it has to be simple and cheap, for both us and the users. This of course can be achieved by using "off the shelf" products. Open source software would also be useful as it can easily be customised for our needs.

To meet all these requirements, the following choices were made. The system should be a simple client-server Flash based one, similar to the ones described in Subsection 2.1.1. These sites are relatively simple to set up and easily customisable. They also allow for a wide user base, as they use a simple web browser and the Flash player, both of which are commonly found on users' PC. These technologies will also typically work through firewalls, unlike other streaming protocols. This was beneficial to us, as it allowed us to stream to restricted users on our university campus.

The Flash player expects the video to be encoded as an FLV file; this can easily be achieved using the open source FFmpeg [FFm08], which again makes this system simple and easy to deploy. One issue with the Flash player was the lack of seeking support, however a *hack* was developed to add this functionality. The exact description of how this hack worked is described in Section 3.2.

We chose a few different genres of content, but for the first round of experiments we served the 2006 FIFA World Cup. This is a hugely popular event with three to four matches on each day. Due to the number of matches each day, many users would miss the live match; this therefore encourages them to use our site to view any matches they missed.

The majority of videos served by our system typically had areas of particular interest, such as goals. To allow users to quickly navigate, we designed and added a *bookmark* feature. While viewing the videos, the users were shown a list of bookmarks to the key events within the content. Then, at any point, the user could click the bookmark to instantly seek to its position within the media. Within our sport content, for example, goals, fouls and similar occurrences were bookmarked.

The concept of bookmarks in media is not new. Most DVDs contain chapter and scene bookmarks, which enable the user to start playback at any location. However, as far as we are aware, there have been no studies on how these DVD features are used. If a user does start a DVD at a specific chapter, it is no technical challenge for the DVD player to seek to the correct point and begin playback. This is not true for video-on-demand, as it can be quite strenuous for servers to seek arbitrarily. As such, the analysis of bookmark use within VoD will be novel.

The rest of this chapter explains how our video-on-demand system was designed and deployed to cater for our interactive experiments, and also outlines the different content used. This also includes the design of different tools to enable seeking within Flash videos.

3.1 System Overview

We set up a simple, interactive video-on-demand system. The system was divided into three main components: the capture server, the Video-on-Demand server, and a web interface as depicted in Figure 3.1a.

Our capture server recorded publicly-broadcasted raw MPEG-2 streams of the programmes selected for our experiments. The recording was done via a digital TV capture device, using VLC [Vid08c] to store the raw stream. Once the full programme had been recorded, the system transcoded the stream with FFmpeg [FFm08]. Two streams were created; high and low bitrate Macromedia Flash 7 FLV files (1 Mbps and 300 Kbps respectively). Administrators would then manually add metadata to the system describing the files. This metadata included the title and description of the video as well as marking the location of key events within the videos which would become *bookmarks* (more details on what was bookmarked is listed in Section 3.3). The final FLV files were then transferred to the VoD server, making them accessible to the users. The full procedure described typically took around twice the length of the recorded video, and so the videos were available shortly after being aired.

The VoD system was an Apache webserver, which served the Flash-based user interface over HTTP. This server was only accessible to staff and students within Lancaster University's campus, and those staff and students connecting remotely via the university's Virtual Private Network (VPN). To aid in logging, all requests made through the user interface were verbose, allowing us to determine exactly which controls users pressed and when. Additionally, each playback window would maintain a periodic (10 second) HTTP-request heartbeat with the server, which was used to determine when connectivity was unexpectedly lost.

To handle user tracking, each user was assigned a unique session ID, which was stored within a HTTP cookie and their URLs. Each event that was logged contained this identifier, allowing us to track individual users throughout their visit to the site. If, however, a user blocked or deleted their cookie, they would appear to be new to the system upon each visit. We note within our analysis where this uncertainty could affect the results.

The web interface consisted of two main sections: an index page allowing the user to select any available video from the system, and the player interface that displayed the video (as shown in Figure 3.1b). We were aware that the user interface would constrain the users' actions somewhat, and it was therefore designed to be as simple and generic as possible. We also wanted the interface to offer modern interactive controls (also called *trick-modes*).

There are many different trick-modes, for example the ability fast-forward or rewind, or the ability to step through the video one frame at a time. However, offering too many trick-modes would clutter the interface, and most of them wouldn't be useful. Therefore, we limited the interface to having just seeking controls (*e.g.* forward, backwards and to arbitrary points), as well as pausing and resuming.

Forward and backward buttons were provided that allowed seeking 10, 30 and 60 seconds in either direction. As these are relatively short distances, we also provided a *seek bar* which enabled users to seek to any arbitrarily chosen time. Finally, a list of *bookmarks* was displayed to the users, which enabled them to jump directly to key events. Bookmarks were added by an administrator, but later the interface was extended to also allow users to submit their own bookmarks (via the *tag* button), which other users could see and use. User bookmarks often covered events that were not typically bookmarked, but were of particular interest (such as events that came under later scrutiny).

3.2 Seekable HTTP Flash

A few tools were created to enable a fully interactive experience in the experiments. Typically, when streaming Flash video (FLV) files from a web server, the full file is streamed start-to-finish, which does not allow for seeking to arbitrary points within the video. Therefore, additional software had to be developed to support seeking.

This software was developed independently in 2006. However, in late 2007, YouTube implemented a system very similar to the one described here. The rest of this section discusses the three main changes which had to be implemented.

3.2.1 Flash Indexing

It is not possible to start playback from any arbitrary byte within a media file, as a media player would have problems decoding the media. As such, an index is typically provided that maps byte offsets to seekable points within the media. The first application which was designed was one which could generate this index.

Typically, stored video is contained within a single file as a long continuous sequence of frames. There is a frame for each picture within the video. Each frame has a unique timestamp, to represent the time at which it should be displayed, and typically these timestamps are at fixed intervals. In Flash video there are two types of frames; key frames and predictive frames. Key frames provide data to generate a full picture, whereas predictive frames provide only the differences since the previous key frame. This allows an efficient way to compress a video, where the complete picture typically does not change every frame.

To play a video, the Flash player must always start at a key frame, otherwise a full picture can not be decoded. Thus, when seeking to an arbitrary point, the player must find the key frame immediately preceding the seek point. An index of key frames to positions within the file must be created to seek efficiently. This index can then be used to find the appropriate key frame when seeking.

For our experiments, software was created to generate these indices. Each index was generated with a custom-made program named FLVTool++ [1]. This C++ program scans through the FLV files, noting the byte offset of each key frame. Once all key frames were found, an index of the timestamps to byte positions was inserted into the beginning of the FLV file as meta data.

[1]Since the release of this software, a product with the exact same name has been released by Facebook [Fac08]

3.2.2 Custom Video Player

Once a FLV file has a key frame index, the Flash video player must be modified to take advantage of this. The Flash player provides a set of APIs which allows simple control over the playback of video. However, it does not provide any control for seeking. Therefore, to provide the appearance of seeking, each time a seek request was issued the Flash player would request a new video stream from the server. This requested video stream URL was in the form of:

http://<host>/play.php?video=<video name>&offset=<offset>

This URL allowed the server to start streaming from a specified offset, and thus the user could seek arbitrarily. The offset sent to the server is the byte offset for the requested key frame within the video file. This position is calculated by the Flash video player using the key frame index contained within the stream's meta data.

Using the URL to pass the offset is not the best way to achieve this. The HTTP/1.1 standard has a *Range* header [FGM+99], which allows HTTP clients to partially request segments of files stored on a HTTP server. The better way to achieve seeking with Flash would be via this Range header, however the Flash player does not support this functionality. If it did, it would simplify processing on the server and aid in caching of the media by traditional web caches.

3.2.3 Server Side Support

As each seek is actually a new HTTP request for a stream starting at a specific offset, there must be some logic on the server which allows the client to begin from any offset within the stream. To achieve this, a PHP script was created which simply opened the file and streamed from the desired offset. This offset was provided in the URL by the client, who found the particular offset using the media's key frame index.

Since the offset points to the beginning of a key frame, an FLV stream header is not present. Since each seek is a new stream, the header is required, as it contains information required for correct playback. Therefore, the PHP script recreates the correct header, and prefixed it to the stream.

Because the video is served as a normal HTTP request, the server will try and transmit the stream as fast as possible. To conserve bandwidth and increase the maximum number of concurrent users, the server was configured to limit the streaming rate. For the first few seconds, the rate was unlimited and then afterwards limited to the video's average bitrate. This minimised the start-up latency and then smoothed playback afterwards.

3.3 Experiment

Over the course of twelve months, this interactive video-on-demand system was used to carry out multiple experiments. These experiments were available to staff and students, and publicised to help attract users. The experiments were run in two phases, firstly covering the 2006 FIFA World Cup[2] and nine months later a wider range of sport and musical events. The content selections were chosen because they had points of interest to bookmark, and would yield sufficient user demand.

The first experiment made a total of 66 matches available from the World Cup (64 from the event itself, and 2 pre-competition friendlies) starting from the 9th of June 2006. Only results after the 13th of June were analysed due to alterations made to the logging system and user interface before that date. Each match was recorded from the beginning of the pre-match commentary through to the end of coverage. At the very least, every goal, penalty, and match start/end-point (inclusive of half-time) was bookmarked.

As a direct result of the first experiments, some new autonomic management techniques were designed. To test this in a real environment, a second experiment was set up. From the 13th of April 2007, we began adding new content to extend the existing catalogue of content. This time, our approach was designed to test the vari-

[2]This is not the only study to look at the 2006 FIFA World Cup, Silverston and Fourmaux took measurements of the PPLive peer-to-peer network as it broadcasted live matches from the event [SF07].

ous new techniques and to revalidate our previous experimental results. Furthermore, we wished to determine the relevance of our analysis/models to other genres (such as music).

Over the following two months we provided the last six matches from the 2007 UEFA Champions League football tournament, some other miscellaneous football matches, seven Formula 1 races, as well as several recordings from music channels and the 2007 Eurovision Song Contest semi-final and final. The football matches were bookmarked in the same manner as the previous World Cup event. In the Formula 1 content we bookmarked the beginning and end of the race, as well as any noteworthy events such as a driver having to retire (after a crash or technical difficulties). Within the musical content the beginning of each track was bookmarked with its corresponding artist and title. A similar approach was taken with the Eurovision Song Contest, where the beginning of each song was bookmarked with the name of the country taking part.

In total there were 88 videos, with an average length of 2.5 hours. The maximum video length was 4 hours, and the minimum length 45 minutes. There were 695 bookmarks, with each video having on average 7.8.

Chapter 4

Analysis and Modelling

As outlined in the previous chapter, two sets of experiments were conducted to characterise a highly interactive video-on-demand system, as well as to test some new content management techniques. In this chapter, we use traces from our system to characterise user behaviour and the resulting workload. Using a combination of *R-Square* fitting and *Kolmogorov-Smirnov tests*, models for the various features were determined. Aggregated results are shown where applicable, but in some cases it is more appropriate to show results for individual videos. We noted in many cases that the features analysed were similar for each video, so for simplicity we will specifically discuss two individual videos in greater detail: the World Cup's Argentina *vs.* Serbia and Montenegro match, and the Eurovision Song Contest final. Both were amongst the top 5 most popular videos and were representative of their genres (namely sport and music). We will refer to these files as *arg-scg* and *eurovision* respectively.

Throughout the two experiments we observed a total of ~1800 unique users to the site, with each video receiving on average 68.2 unique users (and an overall maximum of 383). During this period we served 925 hours of video, which equates to 3.3 terabits of data. We received an average demand of 287±31 requests per day, with Thursday being the most popular. Throughout the day we saw the typical diurnal sinusoidal access patterns averaging 12±10 requests per hour, reaching its peak at midday with an average demand of 29 requests per hour.

We witnessed 123 unique users for *arg-scg*, and 131 unique users for *eurovision*, who watched for a combined total of 29.1 hours and 79.6 hours respectively. Note that if an individual does not maintain the same HTTP cookie between sessions (*e.g.*, their cookie is deleted) they will appear as a new unique user. Equally, if two individuals share the same cookie, they will appear as a single unique user. While we expect these cases to be rare, they may however introduce error into the unique user count.

An observant reader will note that the most popular video had 383 unique users, yet the analysis is concentrated on *arg-scg* and *eurovision* with only 123-131 unique users. The reason for this is that the most popular content, a collection of 'cheesy' music videos, had a very short average session duration. The video was the newest content on the site for many weeks, as such was at the top of the list of videos. We speculate that newcomers to the site would click on this video to understand what the site had to offer, but quickly stop. Shortly afterwards they would continue to explore the other videos on the site, which were perhaps better to their liking. These shorts views were therefore not representative of a typical viewing session and thus the analysis does not concentrate on them.

The rest of this chapter uses the traces obtained from the experiments and characterises the observed user behaviour. Common metrics are looked at, such as popularity, longevity, session length, *etc*. However, to describe the more interactive aspects, new analysis techniques and metrics were developed such as seek distance, hotspot length, jump plots and sequence graphs.

4.1 Probability Distribution

Throughout this chapter different metrics will be fitted to and modelled by different probability distribution. This section quickly outlines the main models, and discusses their uses and relevance.

Normal $pdf(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

A normal distribution is perhaps one of the simplest probability distribution also known as the Gaussian distribution, and recognised as the bell curve. Any variable which is the sum of multiple independent identically-distributed factors is likely to be normally distributed. This ensures that values are centered around a mean with a equal variance either side. There are many examples of normal distribution in natural, for example height of people or the intensity of laser light.

Other distribution are generally not centered around a mean and instead skewed. This is particularly common when mean values are low, variances large, and values cannot be negative [LSA01]. The follow distributions are all skewed.

Log-normal $pdf(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$

Log-normal is a continuous distribution in which the logarithm of the variable is normally distributed. For example, if X is a random variable with a normal distribution then $Y = e^X$ is a log-normally distributed. This distribution is generated by multiple independent variables in a similar way to a Normal distribution however, the variables are multiplied instead of added. Within video-on-demand analysis log-normal can be used to model the size of frames within a video stream, or in some cases the popularity of content [CKR$^+$07].

Exponential $pdf(x;\lambda) = \lambda e^{-\lambda x}$

Exponential is a simple continuous distribution which models the wait times between events, if events occur continuously and independently at a rate of λ per unit of time. Typically it is used to measure the time between particle decays in radioactive materials, or the time between phone calls. More specifically it can model the time between viewing of a video.

Weibull $pdf(x;\lambda,k) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$

This is a particular flexible continuous probability distribution which can mimic the behavior of other statistical distributions such as the normal and the exponential. For example, when $k = 1$ the distribution is identical to an exponential distribution, and when $k = 3.4$ it resembles a Normal distribution. Weibull distributions are commonly used in survival analysis, reliability engineering and failure analysis, amongst others. In VoD it has been used to model the session times, as well as metrics more commonly used by an exponential distributions.

Poisson $pmf(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$

is a discrete distribution which expresses the probability that a number of events will occur in a fixed period of time when events occur independently of each other at a known average rate. This is typically used to model arrival distributions in multimedia systems such as the number of times a video is accessed in a particular period.

The following distribution are considered power law distributions. These are ones where the frequency of an event is proportional to its rank, *i.e.* the i^{th} most popular object receives $1/i^{\alpha}$ of requests.

Pareto $pdf(x; k) = \frac{k\, x_{\mathrm{m}}^k}{x^{k+1}}$

is a continuous power law probability distribution more simply known as the Pareto principle or the “80-20 rule”. It states that 80% of the distribution’s weight is from only 20% of the values. This has been observed in many examples, such as, the distribution of wealth or the distribution of file size in TCP transfer over the internet, *etc*.

Zipf $pmf(k; s, N) = \frac{1/k^s}{H_{N,s}}$

Zipfian distributions have become very popular in computer science, and may be thought of as a discrete counterpart of the Pareto distribution. It is used to model distributions where there are many large rare events, and many small common events. For example, the popularity of websites can be modelled by Zipf as there are millions of websites, which receive only a few users a day; and then there are a few large very popular websites. Zipf has also be shown to model the size of cities (there are a few mega-cities, but many small town) or used to model the frequencies that words occur, *e.g.* words such as 'and' and 'the' that occur very frequently, but many which occur rarely.

4.1.1 Fitting

Throughout this Chapter the different evaluated metrics will be fitted to different mathematical models. There are a couple reasons to do this. Firstly, by fitting the raw data to a model it can aid in understand the shape and implications of the data. Secondly, by creating models it allows the models to be used for future simulations and experiments where using the raw data alone would not be suitable. Both of these reasons aids in the design and development of new algorithms and techniques, some of which are described in Chapter 5.

To test how good a models fits to the raw data two statistical tests were employed.

R-square R^2 The R-Square is a very simple and common indicator of goodness of fit. It works by calculating the sum of the errors between the observed value and the value predicted by the model. More clearly it is defined as:

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_E}{SS_T} \tag{4.1}$$

where SS_R, SS_T and SS_E are defined as:

$$SS_R = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2, \quad SS_T = \sum_{i=1}^{n} (y_i - \bar{y})^2, \quad SS_E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.2}$$

and the variables are defined as:

- y_i is the observed value at x_i
- $\hat{y}_i$ is the value given by the model at x_i

- $\bar{y}$ is the mean value of $y_1 ... y_n$
- n is the number of values.

The calculated R^2 value should be between 0 and 1, indicating how good a fit the model is to the real data, where 1 is a perfect fit, and 0 is a terrible fit.

Kolmogorov-Smirnov (K-S tests) The Kolmogorov-Smirnov takes a different approach, instead it compares the empirical distribution function (ECDF) of the data with the cumulative distribution function (CDF) of the model and simply measuring the largest distance between the two functions. The smaller the value the better the fit.

$$D = max_{i=1...n}(CDF(y_i) - ECDF(y_i)) \tag{4.3}$$

By using a combination of R^2 and K-S tests it is possible to mathematically decide how well a parameterized model fits the real data. The remainder of the chapter will utilise these techniques to explain how well the models fit, and why they are suited to each particular metric.

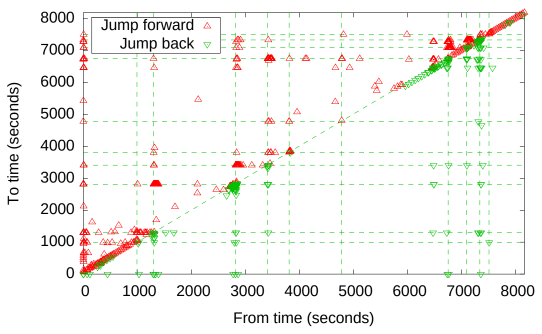
4.2 Interactions

Recall that our system allowed various interactive operations, namely pausing, resuming, seeking forwards & backwards, and jumping to bookmarks. This range of operations, combined with the nature of the content, highly influenced user behaviour. As a result, for most users we observed a complete departure from the typical start-to-finish playback model that has been noted in previous work [CCB+04].

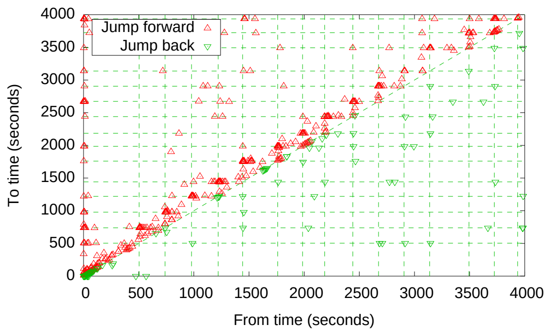
Table 4.1 shows, over the duration of the experiment, the frequency of each action and its corresponding percentage against all other operations. Small individual forward seeks were used a combined 24.9% of the time, whereas individual backward seeking was only used 7.67%. These actions only accounted for the short-seeks

Action	Frequency	Percentage (%)	Mean & Std. (σ) per Session
Back 10s	3098	4.50	0.59 ($\sigma = 3.14$)
Back 30s	654	0.95	0.12 ($\sigma = 0.83$)
Back 60s	1532	2.22	0.29 ($\sigma = 1.90$)
Forward 10s	7438	10.79	1.41 ($\sigma = 8.61$)
Forward 30s	1804	2.62	0.34 ($\sigma = 2.93$)
Forward 60s	7930	11.51	1.50 ($\sigma = 7.38$)
Seek-bar	9902	14.37	1.88 ($\sigma = 7.39$)
Bookmarks	13857	20.11	2.62 ($\sigma = 2.63$)
User bookmarks	1236	1.79	0.23 ($\sigma = 1.01$)
Pause	11839	17.18	2.24 ($\sigma = 7.65$)
Resume	9616	13.96	1.82 ($\sigma = 6.80$)

Table 4.1: Interactions observed throughout the experiment



(**a**) Argentina *vs.* Serbia and Montenegro



(**b**) Eurovision (cropped at 4000 seconds for clarity)

Figure 4.1: Jumps made by users within two videos

buttons (10, 30, and 60 seconds), whereas potentially large seeks (seek-bar and following bookmarks) made up 34.5% of all operations. The table also shows that in each session (a viewing of a single video), a user on average used backward actions once, bookmarks and seek bar actions 4.5 times, and forward actions 3.25 times.

Previous studies have shown that the most common action is pause/resume [CCB+04], however we see that for our traces, forward operations are by far the most common, closely followed by seeking to bookmarks. The table also shows that the number of pause operations account for 17.18% of all actions. Pausing not being the most common action can be explained by the short session durations observed. This is in accordance with previous work which found a positive correlation between session time and the number of pause operations [VPG+05].

To better understand how users navigated through a bookmarked video, we analysed the behaviour in the *arg-scg* and *eurovision* videos, which had 10 and 24 bookmarks respectively. In Figure 4.1a & Figure 4.1b each point is a seek that is identified by a "from" time on the x-axis and a "to" time on the y-axis. A point x,y therefore represents a user that has jumped from their current playback point x to a new point, y. Vertical and horizontal lines in the figures denote the position of the bookmarks. The diagonal line is a current-time marker such that seeks forward are points which lie above it, while seeks backward appear below it. Therefore, no point can fall precisely on the diagonal. It is immediately obvious from the figures that many points are on horizontal lines, implying that most seeks were to the bookmarks.

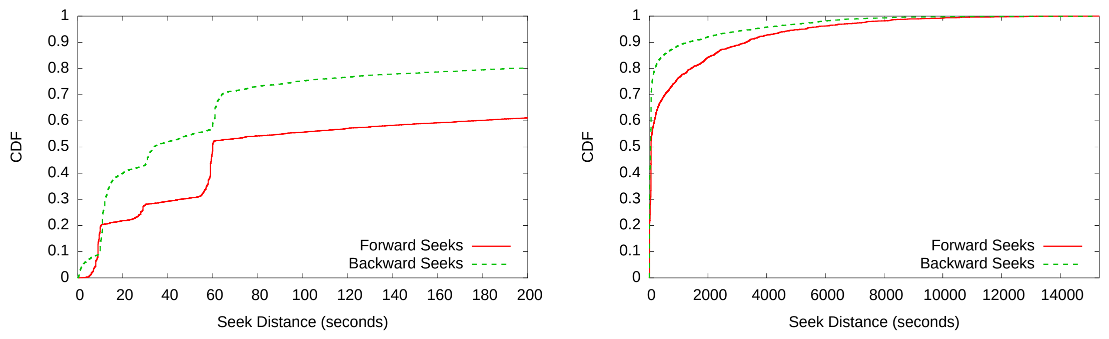
The forward seek buttons appear to have been mostly used for skipping to the next event, shown on both figures as points slightly above the diagonal line between the bookmarks. This could be due to user unfamiliarity with the bookmark interface, or possibly users simply browsing the video. Backward actions were typically used around bookmarks, where users would often re-watch the bookmarked event. In some cases users may also have wished to see video immediately preceding the bookmark. An example of this is shown in Figure 4.1a before the bookmark at time 2815, where users sought up to 75 seconds backwards to see more of the build up to the goal.

Clusters of points can also be seen on horizontal lines shortly after a vertical line, indicating that users jumped from bookmark to bookmark. In fact, the concentration of clusters of point just above the diagonal time reference indicates that users have a tendency to follow bookmarks in sequence, as exemplified in Figure 4.1b.

Overall, for both videos these results demonstrate that users did not simply view continuously start-to-finish, and were in fact highly influenced when presented with bookmarks.

4.3 Seek Distance

The understanding of locality is important for caching and pre-fetching algorithms. By looking at how far users sought we can determine the probability of accessing



(a) All seek distances **(b)** All seeks cropped at 200 seconds

Figure 4.2: CDF of backward and forward seek distances

media nearby the playback point. We therefore define *seek distance* as the absolute difference, in seconds, between a user's current playback point and their requested seek destination.

Figure 4.2a & Figure 4.2b display a CDF of seek distance for backward and forward actions. A large proportion of seeks (between 50%-70%) are of a 15 seconds, 30 seconds, or 60 seconds values. These seeks represent the short seek button presses. 40% of backward seeks were less than or equal to 15 seconds in length. This property could be exploited by keeping a small client side buffer of previously watched segments, which would satisfy many backward seeks if the user has already viewed them.

Even though small seeks are the majority, there are between 30% and 50% of seeks which are further than 60 seconds. These seeks consist of jumps to bookmarks or "blind" seeks with the seekbar. These long range seeks are log-normally distributed with a mean of 1968 seconds and 1630 seconds for forward and backward seeks respectively. They can be fitted to log-normal models with parameters $\mu = 6.8269$ and $\sigma = 1.5953$ for forward seeks, and $\mu = 6.3273$ and $\sigma = 1.7906$ for backward seeks. It can been seen that the backward distribution has a greater positive skew than the forward distribution, thus it will generate many small seeks.

These behaviours exhibit a high degree of spatial locality, with the majority of seeks being within 60 seconds. Regarding long-ranged seeks, the log-normal distribution

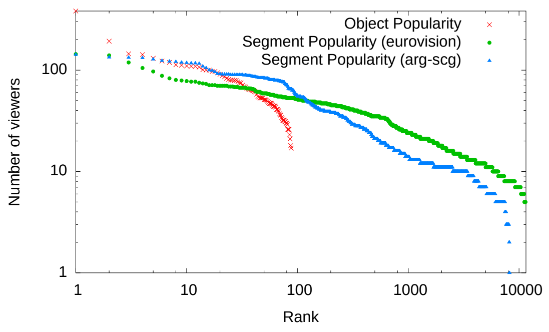


Figure 4.3: Object and segment popularity

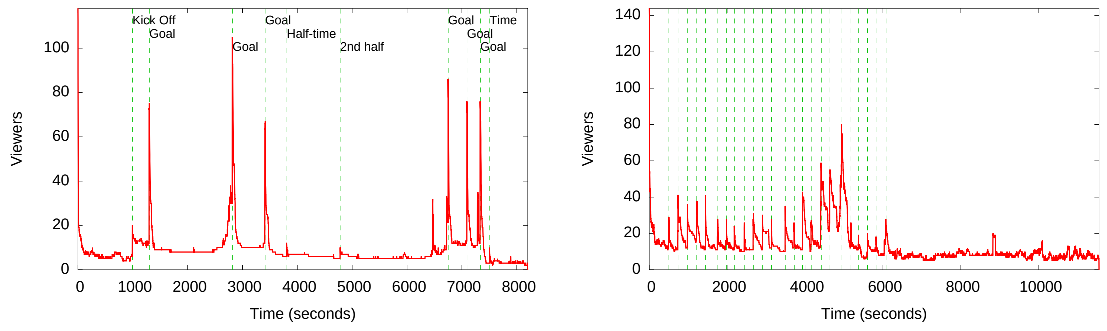
models imply that some very large distance seeks do occur, but the majority of seeks are shorter. Additionally, the skewed nature of this distribution is most likely because it is impossible to have a negative seek value. Overall the seek distances exhibit a median of 60 seconds for forward seeks and 34 seconds for backward seeks. This is consistent with previous findings [PK99].

4.4 Popularity

We study popularity in terms of the number of viewers who watched an *object* or a *segment*. An object in this system is a single video whereas a segment is a section of video one second in length.

The ranking for both object and segment popularity is shown in Figure 4.3. The *eurovision*, and *arg-scg* were approximately 10,000 seconds in length, causing ~10,000 segments to be ranked for each video. Recall that only 88 videos were available, so the lowest object rank is 88.

Typically object popularity with CDNs and VoD systems follows a power-law distribution [CWVL01, AKEV01, YZZZ06], however, our analysis reveals otherwise. Instead the ranking of objects best fitted a normal distribution with parameters $\mu = 60$ and $\sigma = 32$. There are two reasons that power-law was not the best fit. Firstly, the catalogue of 88 videos was not very large, and secondly, power-law distributions do not fit well if the objects are constantly changing. Instead power-law fits better if a



(a) Argentina vs. Serbia and Montenegro (b) Eurovision

Figure 4.4: Number of viewers at each second of video (each vertical line represents the position of a bookmark)

snapshot of rank *vs.* popularity is taken each day and aggregated.

Again, the popularity of one-second segments might be best suited by a power-law, however Zipf and Pareto did not fit well. Instead, the popularity of one-second segments for all the videos exhibit a Weibull distribution with parameters $\lambda = 2.887$ and $k = 0.69527$. Log-normal distributions provide the best fits for the *arg-scg* and *eurovision* results independently with parameters $\mu = 2.00, \sigma = 0.587$ and $\mu = 2.32, \sigma = 0.567$ respectively. Note that log-normal and Weibull distributions closely relate to power-law or heavy-tailed distributions [Mit04, FA06]: they are skewed distributions where a small percentage of samples contributes to a sizeable weight of their distribution. We observe that a small percentage, (the 10% most popular segments), accounted for about 44% of all requests. Previously, Costa *et al.* [CCB+04] found that for educational and entertainment content, the popularity of segments is roughly uniformly distributed with a slight skew towards the beginning for entertainment content. Our result, however, implies that there are segments with orders of magnitude more viewers than others.

To illustrate the order-of-magnitude differences in viewers, we present Figure 4.4a & Figure 4.4b which show the popularity of each second of video for *arg-scg* and *eurovision* respectively. The vertical lines signify the position of the bookmarks; note for the *eurovision* video there were no bookmarks after 6000 seconds as only the per-

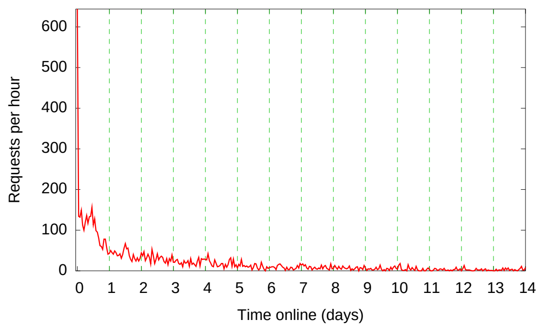


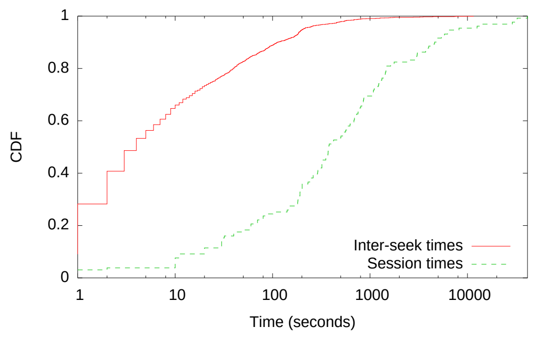
Figure 4.5: Bookmark utilisation within all videos over time, following initial usage

formances were bookmarked and they all appeared in the first half of the video. It is clear from the figures that there are peaks of popularity, highly influenced by the bookmarks. In *arg-scg* (and in other sport content) we observe that most of the bookmarks are equally popular. However, in the *eurovision* (and other music genres), we observe there is a greater variance in the popularity of the bookmarks. This can be attributed to sports having numerous events which all users wish to watch, however in music videos there may be only certain artists which interest the user.

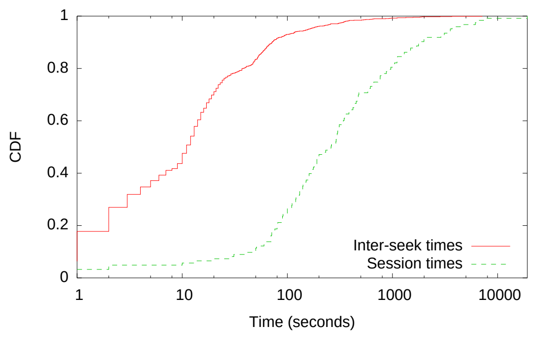
Popularity metrics are important to many CDN algorithms as they help to decide which resources to allocate to each object. We have seen that bookmarks within videos cause segments to be of high interest and popularity, for example, goals within a sporting event. This result emphasises the use of partial caching techniques [CSWZ03] to cache only popular segments.

4.5 Longevity

The popularity of both videos and bookmarks in our system changed over time. This phenomenon is outlined in Subsection 2.2.1.2 which describes how and why the popularity changes. However, in our results the popularity always declined, therefore we call the duration at which any such item remains utilised its *longevity*. The study of a video or bookmark's longevity can aid cache replacement policies, as well as other content management decisions.



(a) Eurovision



(b) Argentina vs. Serbia and Montenegro

Figure 4.6: CDFs of session lengths and inter-seek times

Figure 4.5 shows the popularity of all our bookmarks versus the time they were first used. The figure suggests that following an initial peak and a slight resurgence, there was a rapid decrease in interest after a short period. R-Square fitting reveals that the bookmark longevity can be suitably estimated using a Weibull distribution with $\lambda = 3.10$ and $k = 0.615$. This suggests that the popularity exhibits long-tailed properties. We also observe that 40% of the bookmark usage occurs within 24 hours, with the remainder slowly occurring over the following weeks. This is in line with the previous research on this topic [CG04].

The popularity of videos decreased over time, but this is not true for the popularity of segments within the videos. For example, the segments which were popular within that video when it was first published were still popular within the video weeks later, long after the video had lost it overall popularity. This was tested on each video by calculating the distribution of segment popularity for the first 50% of requests versus the last 50% of requests. The difference in distributions was minor, with an average R-Square value of 0.9. On a visual inspection of the number of viewers per second, it was clear that the popularity still focused around the bookmarks.

4.6 Session Lengths

Session length is the total time a user accessed a video, regardless of the actions they may have taken whilst doing so. For example, a session may be longer than the actual length of the video if the user chose to re-watch segments, and/or pause.

Figure 4.6a & Figure 4.6b show the CDF of both session and inter-seek times (discussion of inter-seek times follows in the next subsection). It can be observed from the session times that most users access each video for a very short time relative to its overall length (possibly just watching the events they are interested in). In particular, note that in the *arg-scg* case around 80% of sessions lasted less than 15 minutes. Given that the video was 2.2 hours in length, 15 minutes corresponds to only 11% of the total video. A similar result was found with *eurovision*, with 80% of sessions lasting less than 12% of the total video duration. The average session duration was found to be only 11 minutes and 18 minutes for *arg-scg* and *eurovision* respectively.

We also found that a small minority (roughly 3%) of session durations were longer than the length of a video. Of these durations roughly 39% were between 3 to 8 hours long. Our logs show that these users paused for a long time before deciding to resume playback.

4.7 Inter-seek Times

Inter-seek time is described as the duration for which a user watched a section of a video before seeking to a new location (disregarding any paused periods). This can be useful, for example, to determine the amount to replicate when using partial caching.

From our logs, we found that on average a user performed 8.98 seek operations around a video, resulting in a mean inter-seek time of 50.4 seconds. Figure 4.6a & Figure 4.6b show the CDF for inter-seek times as well as session length. As the inter-seek times are generally shorter than session times, this implies that the majority of users viewed the content as a series of excerpts, usually under a minute in length.

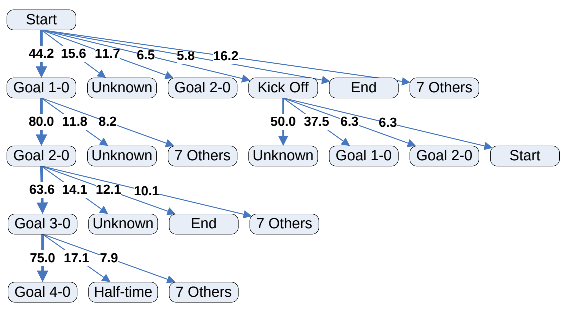


Figure 4.7: Sequence diagram for Argentina vs. Serbia and Montenegro depicted as a tree

The inter-seek time in the music content was found to be on average longer. This is because the length of a bookmarked musical performance generally exceeds the length of an event within a football match. Regardless of the difference in inter-seek times, we found that they can be estimated by log-normal distributions. For instance, the inter-seek time for *arg-scg* can be modelled with parameters $\mu = 2.15$ and $\sigma = 1.72$.

Previous studies have found that the majority of inter-seek times are very short [VPG$^+$05]. For long educational content, inter-seek times have also been shown to be Weibull distributed or a combination of Weibull for the body and Pareto for the tail [AKEV01]. We found that most of our videos had inter-seek times that could be suitably modelled by a Weibull distribution, and two thirds which could be modelled with Pareto alone. Models of inter-seek times can be used by a delivery system to determine the size of video replicas and the time available to react before a user seeks elsewhere in the video.

4.8 Sequence

The traces were analysed to study the extent to which users' actions could be predicted. Since jumps to bookmarks made up a relatively large percentage of all requests, we limit this prediction to which bookmark will be visited next. If a system could predict which bookmark would be requested next by a user, then it could pro-actively respond in order to optimise content delivery. For example, based on the next predicted bookmark, the relevant segments could be pushed out by a server with spare

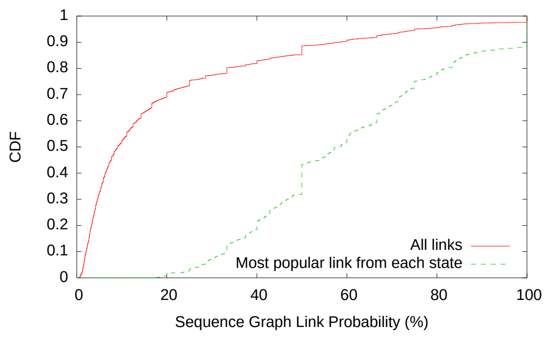
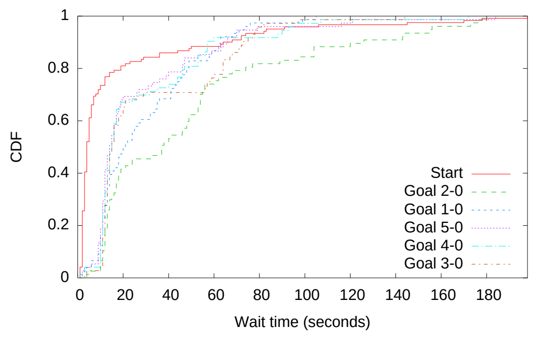


Figure 4.8: CDF of link probabilities for all videos

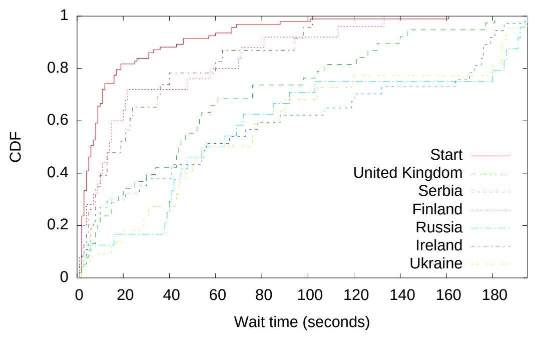
capacity, or pre-fetched by a client.

We call the order that bookmarks are viewed by a single user a *sequence* of bookmarks. Every user's sequence can be aggregated together to form a directed graph. Each node in the graph represents a bookmark with links between them representing the probability of seeking to that bookmark next. Figure 4.7 shows a section of one of these directed graphs depicted as a tree for clarity. The "Start" node represents the beginning of the video, and the "End" node represents the completion of a session. There is also an "Unknown" node which signifies when a seek to another bookmark has not been made within 200 seconds of visiting the previous bookmark (the observed upper bound for bookmarked events' length). For clarity, links with low probabilities have also been aggregated to form a "N Others" node, where N is the number of aggregated links.

It is clear from the figure that there are multiple choices to visit from each node, although there is generally one link that is significantly more likely to be chosen. For example, the probability of viewing bookmark "Goal 2-0" immediately after "Goal 1-0" is 80%. We can also see that following the "Kick Off" bookmark 50% of users did not visit another bookmark within 200 seconds and instead continue to watch, this could indicate that this subset of users were interested in watching the full game instead of just the highlights. An interesting observation for caching is the occurrence of self-loops. 6% of links were between the same two bookmarks, which made up 6.5% of all requests.



(a) Argentina vs. Serbia and Montenegro



(b) Eurovision

Figure 4.9: CDFs of wait times

To understand how many bookmark-to-bookmark links are predictable, Figure 4.8 shows a CDF of probabilities for all links for all videos, as well as probabilities for just the most popular link from each bookmark. From this figure we can conclude that 10% of all links have more than a 58% chance of being followed. Looking at just the most popular link from each bookmark we observe that over half of the bookmarks have an outgoing link with a probability over 50%; an encouraging result for user predictability.

In this analysis we assumed that all users will visit the bookmark in similar order, however in a large heterogeneous environment this may not be true. Different sub-groups may wish to view a different set of events possibly in a different order to other sub-groups. Across our videos we did try and identify if there were groups of individuals that behaved differently to the majority, however none were found. This could possibly be due to our genre of media, with all sports fans wishing to see the same events, in the natural sequential order.

4.9 Hotspot Length

Jumps to bookmarks comprised roughly 20% of all requests with an additional 32% of seeks being within 60 seconds of a bookmark. Bookmarks form the majority of requests within the content, and represent the beginning of a popular segment of

Metric	Distribution	R-square
Object Popularity	Normal ($\mu = 60.129$, $\sigma = 32.111$)	0.97996
Segment Popularity	Log-normal ($\mu = 0.551$, $\sigma = 1.32$)	0.98084
	Weibull ($\lambda = 2.887$, $k = 0.69527$)	0.98284
Session Length	Log-normal ($\mu = 4.73$, $\sigma = 1.90$)	0.99779
	Weibull ($\lambda = 233.17$, $k = 0.51125$)	0.98666
Inter-seek times	Log-normal ($\mu = 1.2886$, $\sigma = 2.318$)	0.99644
	Weibull ($\lambda = 7.5243$, $k = 0.35646$)	0.99353
Seek Distance (forward)	Log-normal ($\mu = 7.2668$, $\sigma = 1.2194$)	0.99567
Seek Distance (backward)	Log-normal ($\mu = 7.195$, $\sigma = 1.3132$)	0.99083
Hotspot Length	Log-normal ($\mu = 2.6361$, $\sigma = 1.388$)	0.98463
	Weibull ($\lambda = 24.594$, $k = 0.7034$)	0.99545
Bookmark Longevity	Weibull ($\lambda = 3.1004$, $k = 0.61592$)	0.99796

Table 4.2: A summary of metrics with their corresponding distributions

video which we call a *hotspot*. The beginning of a hotspot is generally known (*i.e.*, the bookmark point), but the end is not. Knowing the length of the hotspot can be useful for numerous tasks such as caching and pre-fetching. We therefore define *wait time* as the time elapsed between a user following a bookmark and seeking.

Figure 4.9a & Figure 4.9b show a CDF of wait times for each bookmark in the *arg-scg* and *eurovision* videos. It can be seen that in the football match the wait times follow a similar distribution, with the majority of users waiting less than 40 seconds (this, for example, could corresponds to the length of a run up to a goal). The *eurovision* results are more varied with average wait times being much longer. This is due to the typical song in the Eurovision Song Contest being 180 seconds in length. Finally, there is a "Start" bookmark listed in both figures: this is the entry point into both videos, and does not correspond to any event.

To better understand the wait times, distributions were fitted. In the general aggregated case a Weibull model fits best with parameters $\lambda = 24.594$ and $k = 0.7034$. For individual bookmarks log-normal and Weibull models proved best in the majority of cases. With these models the upper bound of a hotspots' lengths can be extrapolated by using, for example, the 95th percentile.

4.10 User Behaviour Models

Model fitting is important for understanding the different properties of the system, and aids in simulation creation and algorithmic design. Various models have been discussed for the different parameters of the system. In all cases many models (*e.g.*, normal, log-normal, exponential, Weibull, Pareto, Poisson, Zipf) were fitted to the data with varying success. Generally, more than one distribution fitted well. This subsection will summarise the analytical models found for each parameter.

Table 4.2 gives an overview of the best matching models for each metric discussed previously, with their corresponding *R-square* values. Of particular importance are the types of distribution which can have a significant impact on the system. For example, the Weibull and log-normal models are both long-tailed, and systems may have to anticipate the skewed distribution to cope effectively.

4.11 Summary

Our results have shown that the interactivity options available to users highly influence their behaviour. In particular, it was found that the novel interactive feature of *bookmarking* played a pivotal role, leading to access patterns quite dissimilar from previous related studies that looked at VCR-like interactivity alone. The combination of our content type and the addition of bookmarks led to users accessing content in relatively short segments sparsely distributed throughout the length of the videos. Segment popularity is skewed with the most popular segments clearly around the bookmarks, forming hotspots. From both a user and a content distribution network's perspective, this can be viewed as advantageous; users can reach interesting content more quickly through the bookmarks, and the increased locality of interest means CDNs can respond more effectively by, for example, prioritising hotspot replication.

Content placement is an important and difficult problem for CDNs. The CDN has to decide where within the network to replicate or cache content. Typically the content is placed near to the users, and replicated as a whole. However, as we have seen, not all segments within a piece of content are equal and a CDN can leverage this information to replicate certain segments more than others. This is especially useful when popularity nearly always concentrates around bookmarks, allowing the relevant segments to be replicated throughout the network before user demand increases.

A CDN could be designed to handle high levels of user interactivity, with relatively short sessions and inter-seek times. Our results have shown that hotspots following bookmarks were orders of magnitude shorter than the video containing them. Furthermore, it encourages the use of an agile delivery mechanism that allows distribution of small sparsely distributed segments quickly and efficiently.

We have also shown that users view the bookmarks in a similar order, giving them a degree of predictability. This could allow a CDN to exploit pre-fetching techniques to improve the user's experience. For example, if the CDN could predict the next segment the user will watch, then this could be pre-fetched into the user's playback buffer and when the user seeks to that segment there will be no delay caused by seek latency and buffering.

The use of bookmarks depends on them being well positioned and of interest to the user. We noted in the first experiment that 40% of bookmarks had at least one user seek before the bookmark, with 30.7% of these seeks occurring within 5 seconds of jumping to the bookmark. This perhaps represents users who were almost immediately dissatisfied with the bookmark's location. We noted this happened consistently for roughly 6% of the total bookmarks. Upon further inspection, it appeared the bookmarks were inadvertently misplaced. This led to users performing additional seeks to find the correct location, thus placing extra load on the servers.

Throughout the experiment different genres of videos were available to the users, namely sporting and musical videos. Only the analysis of the "Argentina vs. Serbia and Montenegro" football match and the "Eurovision song contest" were shown in this chapter, however other sporting events were available on the site such as Formula 1 racing, International Cricket, and other miscellaneous recordings of music channels. Similar patterns were observed for each video, however, semantics of the content did have some impact of how the users consumed the data.

All videos exhibited similar patterns, for example, popularity was generally centered around the bookmarked segments, and that the viewing duration was far shorter than the full length of the video. However, minor differences were found, for example, the music channels had greater variance in the popularity of each bookmark (which were placed at the beginning of individual music videos). This can easily be attributed to users only being interested in particular artists or videos, whereas viewers of sporting events would be interested in every highlight (and therefore every bookmark). Similar differences were found in the inter-seek times, session times, and hotspot lengths, as the semantics of the content would determine how long particular hotspots were. However, metrics such as the number of interactions, or bookmark longevity stayed the same, as these did not appear to be directly impacted by the content.

In the following chapter, we explore and study the implications of a few techniques designed to exploit some of the properties suggested from our analysis. The first addresses the dynamic re-positioning of bookmarks in response to user behaviour. The second concerns predictive pre-fetching of popular segments to enhance the efficiency of delivery of highly interactive content. The last technique is an evaluation of how well existing delivery mechanisms behaviour when delivering interactive media, and how this can be improved with an hybrid approach.

Chapter 5

Techniques for Interactivity Support

So far, this thesis has outlined a twelve month experiment in which highly interactive user behaviour was recorded. The traces obtained from the experiment have been analysed and characterised to produce a set of models and user workloads. These workloads and models were developed so future ideas and concepts could be designed and tested with realistic data.

This chapter outlines some of the improvements which can be made to aid in the delivery of this genre of content. This includes a system to dynamically position bookmarks within the media, a way to pre-fetch segments of the media ahead of their request, and an evaluation of hybrid delivery technique designed to deliver highly interactive content.

5.1 Dynamic Placement of Bookmarks

During our experiments, bookmarks were appropriately positioned by administrators before the video was published. It was previously noted that a small percentage of bookmarks (roughly 6%) were unintentionally misplaced. There are many reasons why a bookmark could be misplaced, such as human error, or a lack of insight into user requirements. For example: a bookmark could be placed before a penalty kick, but many users may first wish to see the foul that led to the penalty. As such, it would

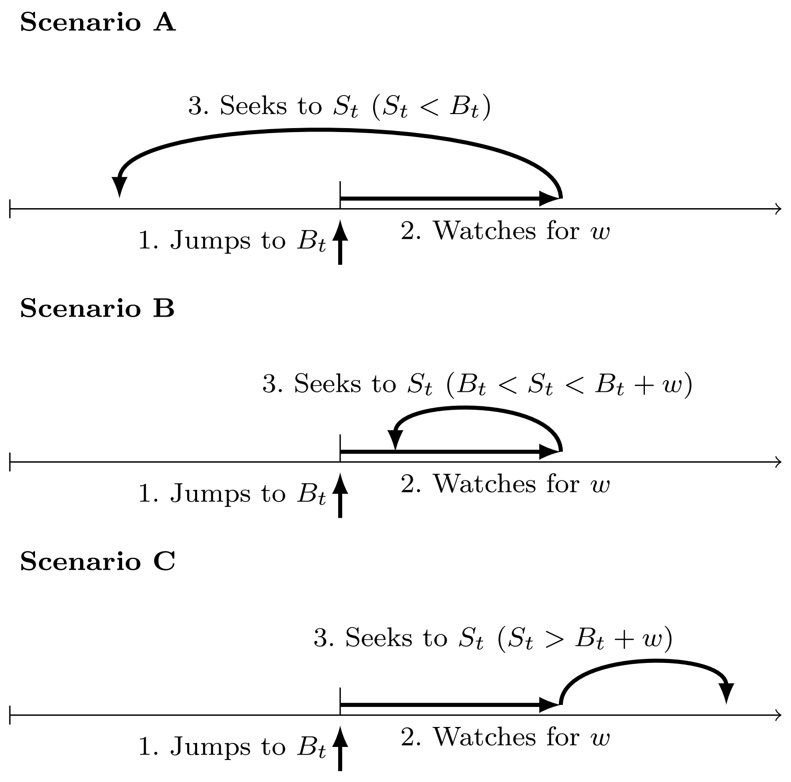


Figure 5.1: Different scenarios that may induce bookmark movement

be beneficial if the system could autonomically detect poorly placed bookmarks and correct them based on automatic feedback derived from the user's actions.

During the second video trial, we took the opportunity to go beyond characterising user behaviour, by testing a dynamic bookmark placement technique in the live system. This technique inferred if the bookmark was misplaced based on the user's seeking behaviour, and then correct the bookmark's position in a reactive way. The remainder of this section discusses and analyse this technique.

To develop a reactive algorithm that moves bookmarks dependent on user behaviour, different possible scenarios should first be explained. Figure 5.1 shows three different sequences of actions a user could follow shortly after seeking to a bookmark.

Scenario A shows the user briefly viewing the bookmark, then seeking to a time earlier than it. While this could indicate that the bookmarked event was short and that the user wanted to view it again, it could equally imply that the bookmark was placed later than it should have been.

Scenario B is similar to *Scenario A* but differs in that the user does not seek back to a point before the bookmark; this means the user is simply replaying footage, thus implying the bookmark is correctly placed for that individual.

Scenario C represents a situation in which the user's motives are difficult to determine. Since they watch briefly then seek forward, several possibilities exist: the bookmarked event may have ended, the bookmark may have been placed prematurely, or the user is simply seeking forward towards the next event.

A further possibility, not shown in the figure, is for a user to seek far away from a bookmark in either direction. Since it is unlikely their destination would be related to the bookmark, such an action would not indicate the bookmark was incorrectly placed.

Scenario A and *Scenario C* are therefore the only scenarios where the user's actions could indicate the bookmark is misplaced. All other actions should reinforce the position of the bookmark to reduce future movements once it is correctly placed. Additionally since we are less sure of the user's intentions in *Scenario C* we should only make minor changes to the bookmark's placement to limit the impact of false-positives.

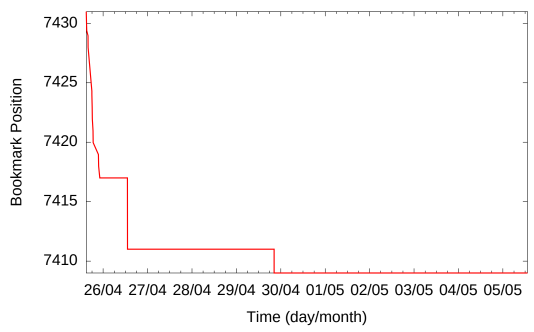
Algorithm 1 has been developed to identify these situations and act appropriately with regard to moving a bookmark. An exponential moving average (EMA) is used to recalculate the bookmark's position with a smoothing constant α. The value used for α is dependent on the identified scenario. Initially these values were 0.1 and 0.05 allowing us to place greater confidence in the seeking-backward *Scenario A* than the seeking-forward *Scenario C*. These values were chosen as the intuitive first guesses for experimental purposes, and should be refined with future experiments. For our testing scenario we also used maximum wait times of 20 and 60 seconds for backward and forward seeks respectively. These maximum values were chosen because they exceeded approximately 80% of all wait times.

Algorithm 1 Dynamic bookmark moving algorithm

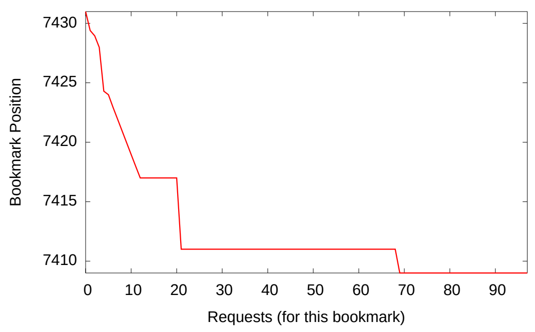
```
// $B_t$ is the location of the bookmark at time $t$
// $S_t$ is the location the user sought at time $t$
// $w$ is the time the user waited before seeking to $S_t$
if $S_t < B_t$ then
    // The user seeks backwards before the bookmark
    if $w <= 20$ and $S_t > (B_t - 60)$ then
        // The seek occurred within 20 seconds of viewing the bookmark and lands within 60 seconds of the
        bookmark
        $\alpha = 0.1$
        $B_{t+1} = \alpha S_t + (1 - \alpha) B_t$
    end if
else if $S_t > (B_t + w)$ then
    // The user seeks forward
    if $w <= 60$ and $S_t < (B_t + 120)$ then
        // The seek occurred within 60 seconds of viewing the bookmark and lands within 120 seconds of
        the bookmark
        $\alpha = 0.05$
        $B_{t+1} = \alpha S_t + (1 - \alpha) B_t$
    end if
end if
```

To test this algorithm, several of the bookmarks in the second video trial (not the initial World Cup experiment) were deliberately misplaced by different amounts before they appeared on the live site. Over time the bookmarks were moved autonomically by our algorithm. For example, Figure 5.2a & Figure 5.2b show the position of a single bookmark as it was moved by the system with respect to time and received requests. In both cases the system responds and the bookmark quickly moves to a new position, and then gradually converges until it becomes stable. In most cases the majority of movements were only in one direction, but for a couple of bookmarks the positions oscillated between two values. The most prominent example of this was a foul in a football match which led to a penalty. Some users wished to see the foul but others only wished to see the penalty a minute later. In these small number of cases it is subjective to decide if a bookmark is correctly placed, and in fact using this algorithm the bookmarks may never converge to a single point. In such cases, it may be best to bias the bookmark towards the earlier position, so both the early and later events can easily be seen.

Instead of subjectively deciding if a bookmark has moved to its correct location,

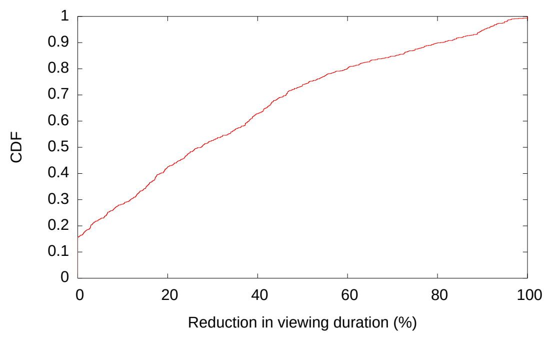


(a) Position over time

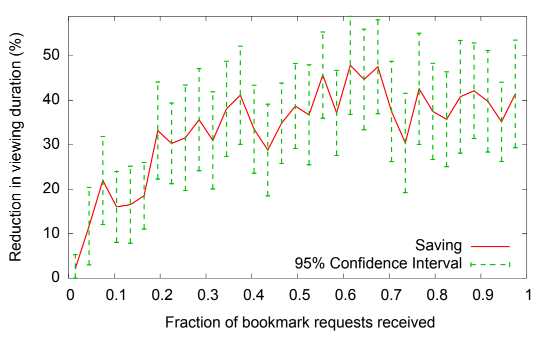


(b) Position over requests for bookmarks

Figure 5.2: Manchester United vs Milan single bookmark position



(a) CDF of percentage reduction in viewing duration



(b) Percentage reduction in viewing duration versus received requests

Figure 5.3: Reduction in viewing duration due to the algorithm

we examined how much traffic might have been saved by moving the bookmark to a new location. If, for example, a bookmark was moved forward 10 seconds closer to the desired location, and a user views for 90 seconds, then by moving the bookmark we have potentially stopped video being transferred, which might have normally been skipped over. A reduction of $10/(90 + 10) = 10\%$ is therefore made. Of course, this is only true if the user does not seek backward to watch the skipped 10 seconds, in which case we save nothing, and in fact incur an extra seek. Figure 5.3a displays a CDF of the potential reduction in viewing duration per bookmark request from the use of the algorithm. We can see that 16% of the requests made no saving: these are accounted for by early requests before the bookmarks were moved, and requests where the user incurs an additional seek.

Figure 5.3b illustrates how rapidly these reductions are made (and whether or not they are sustained) through a plot of the fractional potential saving versus the number of requests received across all the moved bookmarks. For the first 20% of requests the reductions are low yet they improve, and then stabilise at a reduction of between 30-40% per request. The 95% confidence intervals are quite wide in most cases (averaging around ± 10 seconds) although this variance is mostly due to differences in playback length and not the 16% of requests with no saving.

With minimal processing this simple algorithm has been able to reposition the bookmarks to more appropriate locations based on observed user behaviour, resulting in consistent traffic reductions. The algorithm can still be improved by fine tuning the α values. Larger values would move the bookmark more quickly at the cost of increasing the probability of incorrect decisions. This investigation has been left for future work.

5.2 Predictive Pre-fetching

In the classic start-to-finish model it is commonplace to simply pre-fetch ahead of the playback point. This reduces the chance of playback stalling due to momentary network problems. However, due to the increased interactivity of users and their departure from the start-to-finish model, it is no longer wise to only pre-fetch ahead of the playback point. As noted in Section 4.8 it is possible to predict which bookmark a user will view next, allowing the client to intelligently pre-fetch content, benefitting both clients and servers.

For the clients, pre-fetching removes seek latency when seeking to a pre-fetched segment, both in terms of incurred network seek latency and also the time taken to buffer enough video for playback. Pre-fetching also helps to avoid buffer underruns under poor network conditions. Similarly, on the server side, pre-fetching can help reduce the peak server load by increasing the load at quieter times with pre-fetching requests, thus making the overall load more uniform.

However, pre-fetching does come with a cost; resources are wasted if a segment is downloaded and never used. Deciding which segments to pre-fetch is therefore an important task. This section gives some background on the concept of pre-fetching, followed by the design and evaluation of a pre-fetching technique for our highly interactive workloads.

5.2.1 Background

Pre-fetching or pre-loading, is the act of requesting content in advance of demand. This allows clients to have the requested media stored locally before needing it. When a client eventually does request the media, it can be served from the local cache, as opposed to requesting it from a server. Pre-fetching can therefore improve the experience for clients, as there is reduced start-up delay, at the cost of pre-fetching content which may never be viewed.

There are a three main forms of pre-fetching. The first is pre-fetching the media before playback. The second is when playing media, to buffer slightly ahead of the playback point. The last is to pre-fetch segments of media while the media is being played. These three forms will now be discussed in more detail.

When the media is fetched before playback has began it is typically called pre-loading. This involves either fetching the full media, or just the beginning segments of the media [PL01]. Pre-loading the beginning segments is typically used to reduce the start-up latency when the media is periodically broadcast [PLM99]. The decision of what to pre-load is normally based on what content is popular, and how much spare capacity the server has. There have been numerous papers detailing the optimal parameters [BNLT08], such as how much to pre-fetch [Pâr01], how many objects to pre-fetch [Pâr02], how much bandwidth to use, what delivery mechanism to use [CT03], *etc*. From the work, it seems clear that the exact parameters are dependant on the characteristics of the media, as well as how the media is delivered. But overall, pre-fetching at least the first few minutes of media certainly provides benefits.

A very common pre-fetching technique is to buffer ahead of the playback point. This requires a small buffer, large enough to contain at least a few seconds of media. Before playback begins, this buffer is filled and then playback may start from the buffer. The buffer smooths out playback, allowing it to be jitter free, as well as hiding temporary network problems, such as lost packets. This kind of pre-fetching is also useful when streaming variable bitrate (VBR) content [RR97]. Typically, VBR content is bursty, causing periods of time where the server is either under-utilised or oppositely, unable to satisfy the demands of its clients. To solve this, each client requests the VBR content at a constant bitrate (equalling the mean VBR rate), allowing the client's local buffer to smooth out the burstiness [BL99].

The final technique is to pre-fetch segments which are likely to be viewed within the content. This is one of the main areas of interest in this section. The pre-fetched segments can be areas within the media which are particularly popular, and are likely to be watched. Popular segments may occur when users selectively seek within the media and do not follow the typical start-to-finish model of playback. As far as this author is aware, there has been no work which will pre-fetch segments of media in this manner.

The closest example of such pre-fetching, is "link pre-fetching" used by some web browsers and proxies [CY97]. This feature will pre-fetch hyperlinks on the web page that the user is currently viewing. In this way, if the user clicks on a hyperlink which has been pre-fetched, the page will load instantly. However, this technique has been discouraged [Dav01], as most of these systems blindly fetch all hyperlinks on the page. This generates additional bandwidth, and may overload the servers [Duc99].

When content is pre-fetched, it is not always stored in the same place. Earlier work assumed that pre-fetched content would be stored on content servers near to the clients [SRT99, EFV99]. More recent work has assumed that clients are using a set-top box (STB) or PC to view the media, both of which may contain a large hard disk or similar storage device. This allows the content to be available even when the client has no connectivity.

Pre-fetching does not always have to involve a server. In the most naïve systems, pre-fetched data is requested directly from the server via unicast distribution. More advanced systems have dedicated pre-fetch channels which periodically broadcast the segments of media which the server deems useful to pre-fetch [CT03]. More recently, systems have been designed to pre-fetch media from neighbouring clients [SLP+06]. One study by Huang *et al.* found that if peers used their spare upload to assist others in pre-fetching, the server's bandwidth could be significantly reduced [HLR07].

5.2.2 Pre-fetching Strategies

The workloads observed in this thesis exhibited sparsely distributed areas of high interest. This areas have been dubbed "hotspots". A sensible pre-fetching strategies would take advantage of these hotspots, and pre-fetch their segments accordingly.

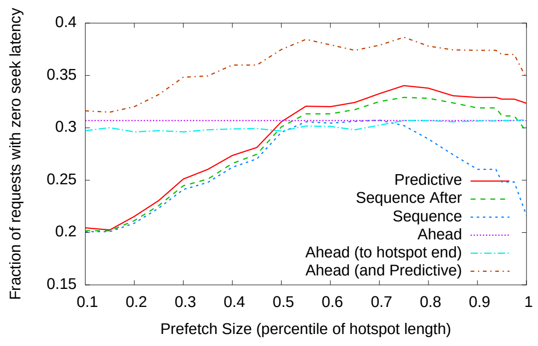
Therefore, a set of pre-fetching strategies were devised. For simplicity, and because interest typical formed around bookmarks, each strategy will only pre-fetch segments immediately following a bookmark (*i.e.*, bookmarked hotspots). In all experiments the amount of each hotspot pre-fetched was determined by varying the percentile of that particular hotspot's length model, as described in Section 4.9.

Each pre-fetch strategies was tested within a simulator driven by the *eurovision* trace. Clients were provisioned with a dedicated link to the server, capable of transferring twice the bitrate required to play the content. Once a client has fetched enough data to fill a 5 second playback buffer, half of their bandwidth is allocated to the pre-fetcher whilst the other half continues to fill the playback buffer.

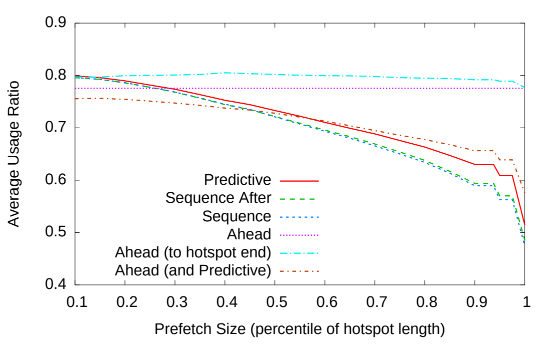
The details for each pre-fetch strategy are listed below:

Ahead simply continues to pre-fetch ahead of the playback point assuming the client has a unlimited buffer. This is similar to what most existing streaming applications do.

Ahead (to hotspot end) again simply continues to pre-fetch ahead of the playback



(a) Fraction of requests with zero seek latency



(b) Usage ratio

Figure 5.4: Various metrics for different pre-fetch schemes versus bookmark length

point but only until the end of hotspot associated with the bookmark being viewed.

Ahead (and Predictive) works in a similar way to *Ahead (to hotspot end)*, however, once it reaches the end of the hotspot it begins to use the *Predictive* pre-fetch scheme.

Predictive uses knowledge observed from other users as to which bookmark is likely to be requested next, and thus starts to pre-fetch the bookmark hotspots in descending order of probability of being visited. This uses the sequence tree concept introduced in Section 4.8, therefore the more users interacting with the system, the more accurate the predictions becomes.

Sequence will pre-fetch bookmark hotspots in the order in which they appear within the video regardless of the current playback point. For example, in a football match the bookmarked goals would be pre-fetched in a sequential order.

Sequence After again pre-fetches bookmark hotspots in the order in which they appear within the video; the difference being only hotspots that are after the current playback point are fetched. For example, if a user has yet to fetch the first bookmark's hotspot but is already viewing the second, then the first will not be pre-fetched.

Two metrics were measured to determine how well the different schemes behaved.

The first metric displayed in Figure 5.4a is the fraction of requests with zero seek latency. A seek latency of zero occurs when the user has already pre-fetched a playback buffer's worth of video from a requested seek point. The second metric measured the ratio of fetched data which was never watched, and therefore needlessly fetched. This usage ratio is shown in Figure 5.4b.

Using the simple *Ahead* scheme 31% of seeks have zero latency, this is made up of seeks to segments that have already been viewed, and small forward seeks into the ahead buffer. Adapting this scheme to only pre-fetch to the end of the bookmarks (*i.e.* the *Ahead (to hotspot end)* scheme) has a minor negative effect on the seek latency, whilst increasing the average usage ratio.

The *Sequence* and *Sequence After* schemes are very similar, but the simple modification to the *Sequence After* scheme allowed it to achieve a lower seek latency whilst not degrading its average usage ratio. This was because users had a tendency to not seek to a bookmark before the current playback point, and always go forward within the video, leaving the *Sequence* scheme stuck pre-fetching hotspots before the current playback point.

Both the *Predictive* and the *Sequence After* schemes perform in a similar manner, with the *Predictive* schemes always outperforming the other. Due to this fact, the *Sequence After* scheme could be used in place of the *Predictive* scheme whilst knowledge is collected to improve the *Predictive* scheme's accuracy.

The best outcome was the combination of *Ahead* and *Predictive* schemes named *Ahead (and Predictive)*. This exploited the fact that users rarely viewed beyond the end of a hotspot, and thus pre-fetching another hotspot was of benefit.

5.2.3 Pre-fetching Knowledge

In the previous experiments the *Predictive* scheme was primed with knowledge from all users, but in reality this knowledge would be built up over time. To test how quickly this knowledge could be obtained, we ran another set of experiments where

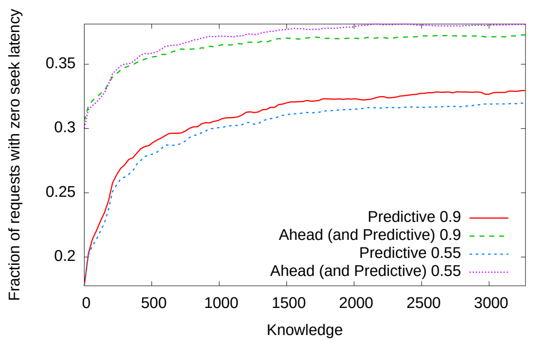


Figure 5.5: How zero seek latencies is effected by amount of pre-fetch knowledge

Media	From (seconds)	To (seconds)	Frequency
arg-scg	0	1024	34
arg-scg	0	271	10
arg-scg	0	0	5
arg-scg	271	1024	8
arg-scg	279	1024	2

Table 5.1: Example links table, showing the frequency of seeks from one time to another

the *Predictive* and *Ahead (and Predictive)* schemes were primed with different amounts of knowledge. The results of this are shown in Figure 5.5. We set the percentile hotspot length to 0.55 and 0.9, which were chosen since 0.55 is where the seek latency began to stabilise, and 0.9 is where the usage ratio began to drop rapidly. The knowledge is ranked from 0 to 3000 which represents the number of seek requests the knowledge was based on. It can be seen that very quickly (within 250 seek requests) the knowledge has become useful, and eventually plateaus at 1500 seek requests. Any seek requests after this point just increase the confidence in the knowledge and does not improve it.

In a real system, predictive knowledge must be collected in real time, and then disseminated in an efficient, scalable, and quick way otherwise any benefits gained may be lost in overheads. One solution to gathering this knowledge is described below.

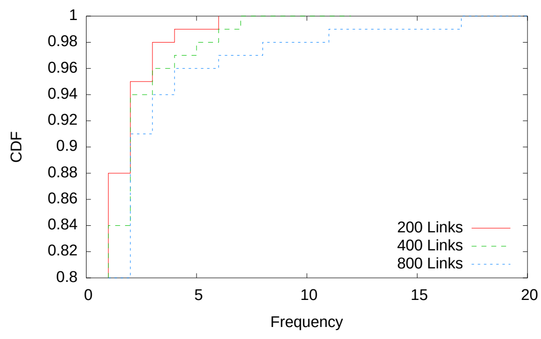
To gather this knowledge, the clients and servers must store a small amount of state. Clients must record what is currently being requested. These details are typically recorded by the client anyway, for example, the name of the media, as well as the location playback started within the media. To make the solution scale, and generate less overhead for the server, it has purposely been designed so that the server need not store per-client state. The servers must only store a table of links relevant to the media stored on that server.

An example of the server's links table is shown in Table 5.1. This table maps the frequency of seeks between two locations within the media. For example, Table 5.1 shows that 34 seeks occurred between time 0, and time 1024 within the *arg-scg* media. This table may be used by the server or clients to predict what they will visit next, for example, if a user has just started playback of *arg-scg*, they have a $34/(34 + 10 + 5)$ or 69% probability of seeking to time 1024 within the media.

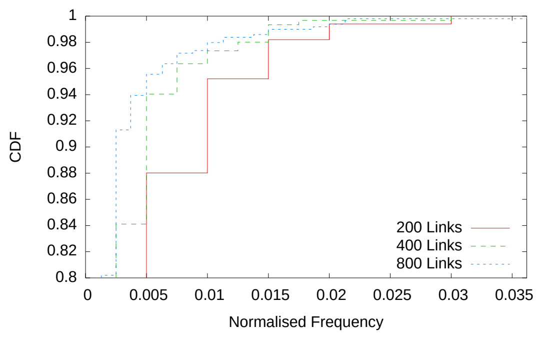
To construct this table the client must send some additional information to the servers. Typically when a client seeks, a new request is issued and the old request is stopped. This new request may be issued to a different server if the media is partitioned across multiple servers, or just for load balancing purposes. So when a client issues a new request, it must send details of the new seek location to the previous server. These details can be sent when the client stops the previous request.

By sending details of the new request, the previous server is able to infer a link between the two requests, even if the next request is supplied by a different server. The server does not need to store any additional per-client state, at the cost of trusting that the client will always send correct information. The inferred links can be stored in a table along with the frequency of their occurrences. This is the beginning of constructing a prediction tree. Once sufficient links have been inferred, a tree can be constructed.

Once this links table begins to be constructed, it can be used by both the servers and clients. When a user makes a request, the server can send a subset of the table to the client as out-of-band data. For example, if the client requests second 0, then the server would send the subset of rows whose *from* time is 0. This would give the client the knowledge to predict which locations are most likely to be visited following



(a) Link Frequency (Absolute)



(b) Link Frequency (Normalised by from time)

Figure 5.6: CDFs of Argentina vs. Serbia and Montenegro link frequencies with varying numbers of links in the table

the current location. As the client continues playback, the server can send additional subsets of the table. For example, if the client continued playback for a further 60 seconds, then subsets of the table from time 60 will be sent, again as out-of-band data.

When multiple servers are used, the links tables can easily be shared or partitioned among the servers. If two servers create the tables independently, they can be easily aggregated to produced more accurate data. The frequency of sharing the links table is left for future work.

The size of these tables may quickly be filled with requests with only a small frequency. Figure 5.6a displays a CDF of link frequencies for one table with varying number of links in it. It can be seen that when there are 800 links, roughly 80% of these have a frequency of one. These small frequencies may not be useful. Therefore, there are a few ways to reduce the size of the table. The first technique will reduce the resolution of the table by grouping similar times together. For example, in Table 5.1, the values 271 and 279 could be rounded down to a single value of 270. Then all matching values can be easily aggregated. This technique would most likely be used to at least round down to the nearest keyframe. As playback of media can only start from a keyframe, then it makes sense it always round the *from* and *to* times to the nearest preceding keyframe.

The second method aims to remove "noise" from the data. Each frequency can be transformed by some operations, such as a division or a right bit shift. Any frequencies which become zero or less would be removed from the table. Any values large enough to be significant will be kept, and the long tail of small results will be lost. A count of how often the table is reduced in this way must be kept to enable the table to be aggregated easily with other servers.

A final suggested technique would use the concept of aging, whereby entries in the table that have not be observed recently are removed from the table. This technique may provide the most relevant entries at the cost of additional overhead for each entry.

An evaluation of these techniques has not been conducted in a real system, and is therefore left for future work. However, the system has been designed to be lightweight and take advantage of how the delivery systems normally operate. The client, for example, only needs to send a small amount of additional information piggy-backed on existing communications. Additionally the servers do not have to handle any "heavy" per-client state, and instead only store the links table, which even if large would only consume 10-50 kilobytes[1] per media object.

5.3 Hybrid Delivery

This section discusses how existing peer-to-peer (P2P) delivery techniques are unable to provide a sufficient level of interactivity to adequately support the workload analysed in this thesis. To improve the performance of P2P delivery, this section discusses a hybrid approach which can use the best features of two main classes of streaming P2P. These two classes are 'pull' and 'push', which have previously been described in Subsection 2.1.5.

To recap, push based systems configure the peers in a tree topology, and disseminate the content efficiently throughout the tree. This however does not give the peer freedom to seek within the stream, as everyone in the tree is at the same playback point. To seek, the peer would therefore need to join another tree, causing long seek

[1]Based on 12 bytes per row, and between 800–4000 rows.

delays.

The pull based system do not typically try to configure the peers into any structured way, instead creating random meshes of connections between the peers. Each peer must advertise the segments of the media they currently have stored locally. Peers may then request these segments arbitrarily, allowing them to play back the media in a sequential manner, or if they wish, seek to any segment and resume from there. This however requires a high amount of overhead as segment advertisements must be continually exchanged and each segment is only received after an solicited request.

The remainder of this section describes a hybrid approach which constructs the peers into multiple push-based trees using a periodic broadcast technique. The peers also form a light weight pull-based mesh, so they can use a P2P patching technique to allow for quick seeking. The following section describes these typically multicast/server orientated broadcasting and patching techniques, and how they relate to peer-to-peer. Following this, design, analysis and evaluation of the hybrid scheme is discussed.

5.3.1 Periodic Broadcast and Patching

It is cheaper in terms of resources to serve ten users via a single multicast stream, then to serve the same ten users via unicast. Creating a single multicast channel decreases network load, as only 1/10 of the bandwidth is needed, and also reduces requirements placed on the the server, such as the memory and disk IO bandwidth. However, it is not always possible to ensure that the users all wish to start watching at the same time; batching was therefore created to solve this [DSS94, AWY96, VI96, HS97].

Batching in its simplest form groups users with the same playback point together and delays them receiving the media until a suitable multicast stream can be found/created to accommodate the full group. This works well if the media is periodically broadcast [Chi95] on multiple multicast channels at fixed intervals apart. This technique is also called staggered broadcasting. When a client joins or seeks to a specific point [BHH$^+$94], they wait until one of the channel's playback points matches their required playback point. However, this causes a long delay, for example, if a 60 minute video is broadcasted over 10 evenly distributed channels, it can take at most 6 minutes to view any requested location, but on average only 3 minutes. To allay this issue, enhancements such as harmonic broadcasting [JT97] or patching can be employed.

Patching (or sometimes called Stream Tapping [CL97]) extends batching by adding a mechanism to remove the start-up delay whilst still utilising multiple dedicated channels [HCS98]. When a client requests a playback location which can not instantly be served by an existing channel, a separate patching channel is created. This can occur when the client first views the content, or if they seek to an arbitrary point. The client then requests both the patching channel and a dedicated channel whose playback point is later than the requested playback location. The patching channel contains the data required to catch up with the dedicated channel and is typically unicast from a content server or sometimes from neighbouring clients [KPS08]. This then allows the client to begin playback instantly from the patching channel and buffer from the dedicated channel. Once the patching channel catches up with the client's buffer, the patching channel is stopped.

The idea was extended to reduce the number of required dedicated channels. For example, consider there are two dedicate channels one being 10 minutes behind than the other. If a client is currently viewing the channel which is behind, they may start to patch from the ahead channel. After 10 minutes they would have caught up with the ahead channel and thus can disconnect from their behind channel. This then might allow the channel to be freed if no other users are viewing it.

The duration for which the patching channel is used is called the *patching window* [CHV99]. The size of the window can impact performance, for example, the *greedy patching* scheme tries to minimise the number of multicast channels by allowing the patching window to be as long as the content's duration. It has been shown that being too greedy can result in less data sharing [HCS98]. The opposite of greedy patching is *grace patching*, which schedules a new multicast channel when the client's buffer is smaller than the patching window.

A middle ground is controlled multicast, which adds access controls limiting how many patching channels can be created. For example, the controlled *CIWP* algorithm [GT99] uses a mathematically optimal scheduling algorithm that limits the rate at which patching channels are created. Another technique, Lambda Patching [GLZS00], allows the server to decide on the patching window sizes, based on currently observed popularity and interarrival times.

A final scheme is *transition patching* which increases the size of the patch window, but allows multiple clients to share a patch stream, thus increasing bandwidth sharing [CH99]. The efficiency of these patching schemes has been measured [BWS$^+$01, EVZ01], modelled [TEVG02] and optimised when considering factors such as video length, client buffer size and request rate [CHV99, EVZ99, SGRT99].

Although patching was originally designed to provide simple video-on-demand, it also enables interactivity. More recently, Ma *et al.* optimised patching for high levels of interactivity [MSW05]. However their evaluation used non-realistic interactivity patterns. Rocha *et al.* tested patching with more realistic interactive patterns and found that patching did not scale well under high levels of interactive requests. With low levels of interactivity, patching performed sufficiently. However, with medium to high interactivity, patching performed in a similar way to serving each client with an individual unicast stream [RMC$^+$05].

5.3.2 Delivery Methods

To compared a peer-to-peer hybrid approach to the existing push and pull methods, three delivery mechanisms are defined and evaluated. This section describes the three mechanisms in detail.

5.3.2.1 Push – Periodic Broadcast over TBCP

The first of the three approaches aims to group similar users together on a multicast substrate. Given the lack of support for network multicast, we use an application-level multicast (ALM) algorithm, specifically the Tree Building Control Protocol [MCH01], which in practice is a protocol to arrange the peers in a push-based P2P tree.

Periodic broadcast can then be used in order to provide the illusion of true video-on-demand. Staggered broadcast is its simplest form, involving a number of multicast channels for a given video, and beginning each at an interval evenly distributed throughout its length [DSST95]. Users may then move forwards and backwards throughout the stream by switching channels. The granularity of these operations is, however, limited by the interval between the available channels. Without allocating a large number of broadcast channels, startup/interaction latency may be unacceptable in longer pieces of media. An adaptation of this method is our push approach to delivering the media, in that content nodes broadcast the same segments of content to as many nodes as possible simultaneously, where TBCP trees are generated as required to act as periodic broadcast channels.

5.3.2.2 Pull – Peer-to-Peer

The second delivery method considered, representative of the pull approach to delivery, is the use of a peer-to-peer, BitTorrent-like protocol. BitTorrent implements a tit-for-tit incentives mechanisms, to discourage free-riders[2]. A beneficial side-effect of involving time-sensitive data in such a network is that the incentives that drive the system become more pronounced. In other words, due to the tit-for-tat exchange mechanism, would be encouraged to contribute, as to improve the probability of receiving the segments they require in a timely fashion. Such protocols are not without drawbacks in the context of streaming media, however.

[2]Peers who chose to not contribute to the network, thus obtaining a service at no cost to themselves.

Firstly, the most common issue is that media playback requires sequential segments of a piece of content; BitTorrent instead delivers them effectively randomly. A solution often proposed to handle the need for sequential delivery is to apply a sliding-window to the data being distributed. Peers then have upper and lower boundaries for the segments which they are interested in downloading, and prioritise their requests accordingly.

Secondly, startup latency should be minimised; BitTorrent's reliance on other peers optimistically unchoking newcomers means this is not the case. Shar *et al.* and Vlavianos *et al.* amongst others have noted and responded to issues such as these [SP07a, VIF06b]. A simple solution is to ensure that established peers perform optimistic unchoking on a more regular basis; Shah and Pâris show that, when combined with a sliding window approach, significant improvements over the base protocol can be achieved [SP07a]. We therefore employ similar methods to gain the same benefits.

5.3.2.3 Hybrid – Periodic Broadcast with P2P Patching

Finally, the third method examined is a hybrid of both the push and pull methodologies. By using TBCP-based periodic broadcast trees in conjunction with peer-to-peer patching, clients can share data enabling them to reach the current broadcast point rapidly. To clarify, when a client joins a periodic broadcast channel, they do not need to wait for channel to reach the desired playback point, but can instead request segments from their neighbours as quickly as possible until the broadcast channel reaches the required position. This system is similar to the one described by Guo and Ammar [GA04]. They used periodic broadcast over ALM, supported by server based patching. They showed promising results, however, their workload was an artificial start-to-finish.

5.3.3 Experiment

Three key variables to be considered for a delivery system are the size of the audience, the nature of the content (workload) and the network resources available. We also vary *peer-to-peer usage*, in terms of the number of connections they are allowed to make to other peers. For example, if in an unrestricted environment there were 200 peers, a value of 10% P2P usage would reduce the number returned to just 20. By varying this value, the effect of peer-to-peer patching on the hybrid method can be observed: at 0% usage, the method is identical to the normal periodic broadcast over ALM, whereas at 100% usage, the peer-to-peer scheme is made use of as much as possible. The increasing effect of the peer-to-peer patching system in the hybrid approach can therefore be observed in an incremental fashion relative to both the pure ALM and pure P2P delivery methods. Naturally, this does not affect the pure ALM approach, as no variance is seen in its results for plots of this type.

To obtain results that provide a comparison of how particular delivery schemes may handle different pieces of content, varied workloads are considered. These can be classified as follows:

Start-to-Finish Used for baseline comparison, where clients do not use any interactivity features. Media is viewed in a start-to-finish fashion, although the start and end points are not necessarily the same between clients.

Interactive This workload is derived from the analysis and modelling conducted in Chapter 4. In essence, the unique aspect of these workloads is large pieces of content with relatively short areas of high interest, highlighting how well a particular approach handles large differences in popularity between segments.

Beyond the workloads, the metrics used are selected to reflect both network provider and user satisfaction, based on resource usage efficiency and perceived quality of service respectively. The first metric considered, therefore, is *network stress* in terms of the amount of data delivered on the network, normalised to the worst case in a given simulation.

The second metric used is 'timely' segments, *i.e*, one which arrived before its playback deadline. An *average fraction of timely segments received per client* is therefore the average value, on a per-client basis, for the total number of timely segments divided by the total number of segments required by that client.

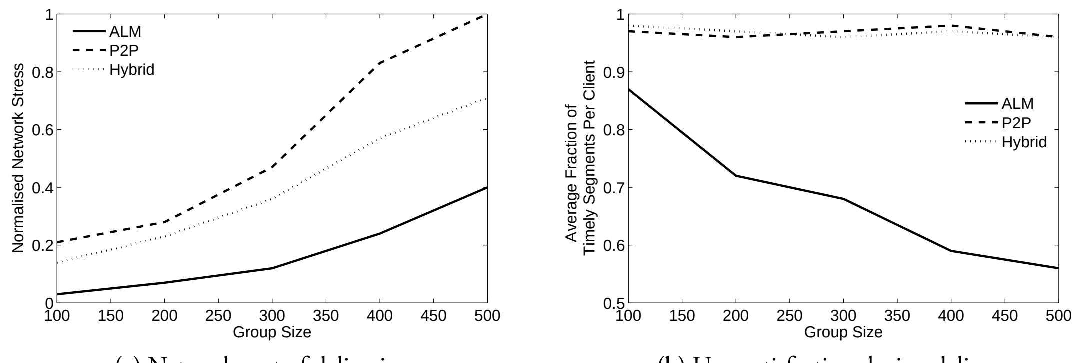
Finally, *segment utility* is considered: the number of media segments actually used in playback in comparison with the total number sent over the network. Given that a client may receive segments superfluous to their requirements (*e.g.*, buffering five minutes when the client only watches one. This metric provides a measure of network efficiency relative to the data that clients actually consume.

A high-level overview of the experimental setup is as follows: 1,000 routers exist in a GT-ITM generated topology graph [CZ]. The graph is of a realistic transit-stub configuration, wherein a single node exists per transit domain, but many exist per stub. Content node(s) (those sourcing the media) are attached to the aforementioned transit node(s), whereas clients are attached to randomly selected members of the stub domain(s). The content node(s) are connected via highly provisioned links, whereas, the clients are limited to a typical asymmetric link (1Mbit down, 256kb up).

The group size used in the simulations is 500 clients, and delivery of some subset of the piece of content per client is made using the appropriate method described for each simulation. Each client possesses an individual playback buffer, which should not underflow if user-satisfaction is to be achieved.

In our custom simulation environment, the overall delivery process is modelled in a number of steps over several iterations, whereas the content itself is handled as a series of sequential 'segments'. Firstly, a realistic workload is generated (according to the models from Chapter 4), wherein the clients are provided with individual lists of segments they will be required to obtain within the timeframe of the simulation.

Following workload generation, clients must interact with the delivery structures to begin the acquisition of segments. In the pull peer-to-peer approach, delivery pro-



(**a**) Network cost of delivering (**b**) User satisfaction during delivery

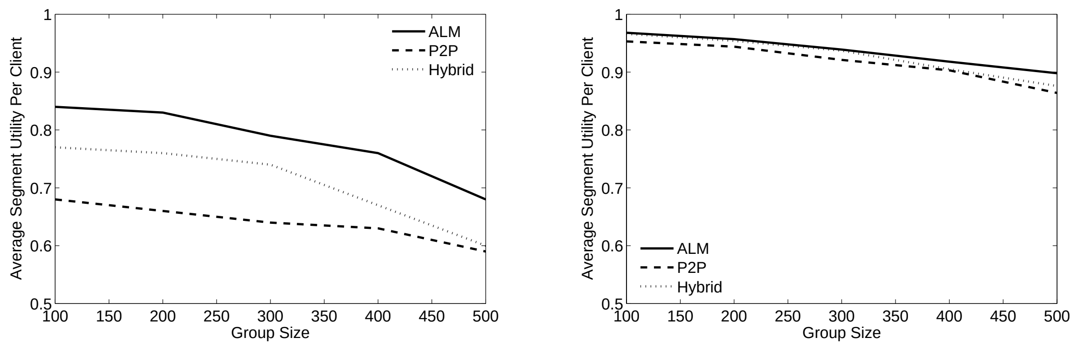
Figure 5.7: Delivering a Interactive workload with varied group size

ceeds as described in Subsection 5.3.2.2. The push and hybrid approaches, however, primarily use TBCP periodic broadcast trees, meaning the initial step is for clients to determine which tree(s) will contain the segment(s) they require, and where they are rooted (*i.e.*, the address to join). In the simulations this is achieved through a lookup process on a single node for simplicity, although in a real-world deployment with numerous videos of many segments each, a more sophisticated arrangement may be appropriate.

Following location of the correct tree for the required segment(s), a client must then wait on the periodic broadcast to roll-around to the required playback point for consumption. In the hybrid approach, the peer-to-peer mechanism can now be used to speed this process somewhat, by attempting to rapidly patch required segments that the client would otherwise have to wait for. Note that clients also have to use the knowledge of segment-tree mappings obtained from the lookup service to anticipate when trees must be switched, as to avoid suboptimal resource usage through nodes being present in a tree unnecessarily.

5.3.4 Results

Figure 5.7a shows the resulting network cost across the delivery methods for an interactive workload with varying group sizes. The amount of traffic created for the ALM-based methods is shown to be lower than pull peer-to-peer; significantly so for



(a) Delivery of an interactive workload (b) Delivery of a start-to-finish workload

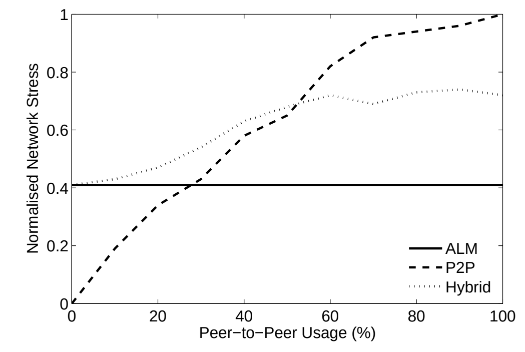
Figure 5.8: Overall segment utility during delivery with varied group size

large groups. A delivery method being low cost, however, is of little use unless it can deliver an equivalent quality of service. Figure 5.7b shows that this is not the case for pure ALM, as an initially high level of timely segments reduces rapidly with increasing group size. Such a result may be indicative of this approach being unable to handle large, interactive groups well, possibly due to the joining overhead associated with switching between highly populated broadcast trees. Indeed, when delivering a 'start-to-finish' workload (not shown), pure ALM was found to provide a level of performance similar to the other methods for the considered metrics.

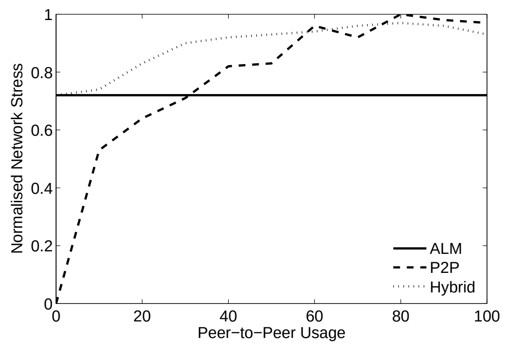
Figure 5.8 show the average fraction of useful segments that were delivered during each simulation on a per client basis. Interestingly, Figure 5.8a indicates that for a high-interactivity workload, up to 40% of segments on the network could be sent fruitlessly, with the factor of interactivity separating the delivery methods noticeably. In contrast, Figure 5.8b shows that when the workload is start-to-finish, each of the schemes achieves a similar level of overall utility, working at an efficiency of around 85% upwards at all times, with little to separate the methodologies. The 'wasted' segments in these cases are most likely due to congestion, and accordingly show an increase with group size.

5.3.4.1 Varying Peer-to-Peer Usage

Figure 5.9a shows the network stress in terms of data delivered per link for an interac-

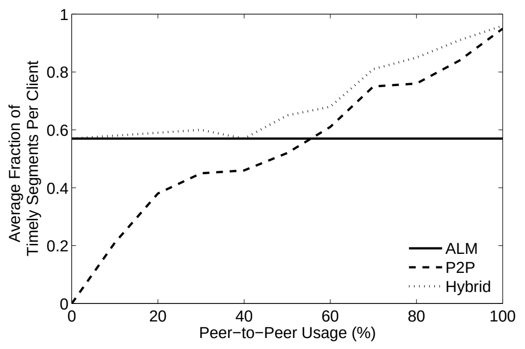


(a) Delivery of an interactive workload

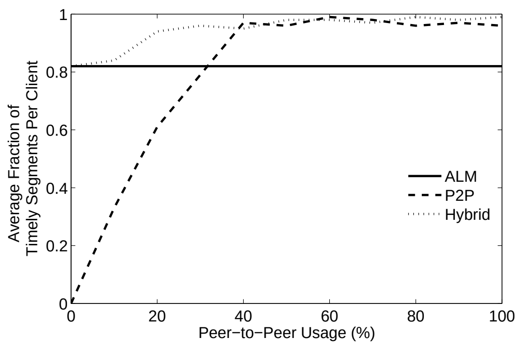


(b) Delivery of a start-to-finish workload

Figure 5.9: Network cost of delivery with varied peer-to-peer size



(a) Delivery of an interactive workload



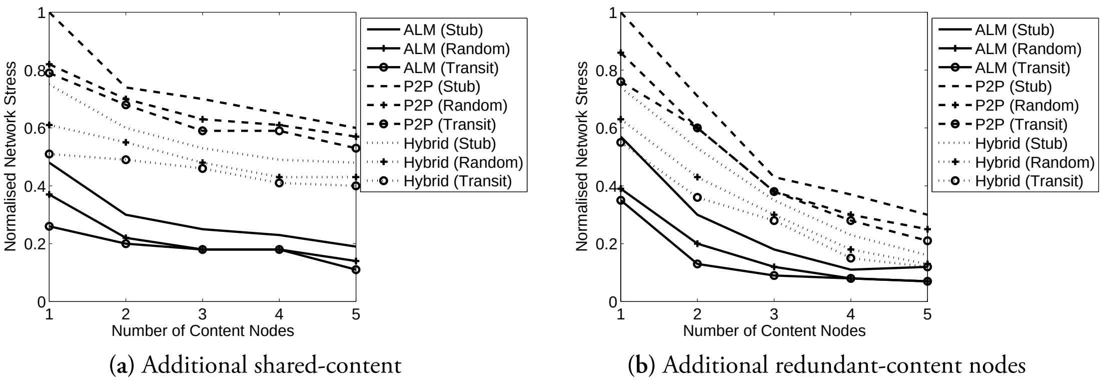
(b) Delivery of a start-to-finish workload

Figure 5.10: User satisfaction during delivery with varied peer-to-peer size

tive workload, normalised to the 'worst case' of full peer-to-peer usage. As simulations run to the point where clients no longer have any interest in receiving segments, the reduced traffic in the low usage scenarios for the peer-to-peer case is to be expected. Two notable points on this plot are around 25% and 50% peer-to-peer usage, where the pull method crosses over the ALM and hybrid approaches respectively. Following the latter intersection, the hybrid method appears to level off while pull P2P continues to increase. Dependent on the user satisfaction for these schemes at these points, this indicates that the hybrid method can produce a consistently lower amount of network traffic relative to push P2P, although the addition of patching is resulting in twice the amount produced by ALM alone. A key point to consider regarding the high network cost of pull P2P is that the ALM-based approach is exploiting the concept of locality in building shortest-path trees to the highly provisioned content nodes. In the pull peer-to-peer case, this does not happen, and many segments are likely being sent lengthy distances between groups of clients viewing the same section of video, potentially across multiple network transit/stub domains.

If the first figures are considered as showing network cost, then the likes of Figure 5.10a show the resultant level of 'user satisfaction', in terms of the average fraction of timely segments per client. Recall that two particular points of interest in Figure 5.9a were around the 25% and 50% peer-to-peer usage marks. In this figure, however, no significant improvement over pure ALM is shown until around the latter of these points: the 50-60% mark. It therefore seems that without the ability to disseminate a large amount of P2P data, the use of the pull P2P approach just results in additional overhead for this type of workload. As the usage increases beyond this point, however, the number of timely segments being delivered increases significantly for both the schemes that make use of P2P. In almost all cases, however, the hybrid method outperforms pull P2P, although not by a large amount. This may be due to the sequential nature of the media playback – when periodic broadcast is used, most of the data that clients need will be pushed to them anyway – the pull approach may only be particularly useful when clients have to seek to new points in the media. The high level of interactivity in the workload used for this particular figure may therefore explain the relatively poor performance of pure ALM.

Figure 5.9b shows the result of a simulation similar to that conducted for Figure 5.9a, but with users consuming the content 'start-to-finish'. The network cost is found to be closer between the three methods in this case than with a high-interactivity workload, supporting the idea that user interactivity can be more costly for certain delivery methods. Interestingly, when examined in conjunction with Figure 5.10a, it can be observed that the pull P2P approach outperforms the hybrid method under these metrics when "peer-to-peer usage" is around the 40% mark. This particular simulation result may therefore run contrary to the initial expectation that



(**a**) Additional shared-content (**b**) Additional redundant-content nodes

Figure 5.11: Effect of additional nodes on overall network stress

periodic broadcast may be better suited than pull P2P for many users viewing the media in a generally sequential fashion. Also note, however, that the average fraction of timely segments for pure periodic broadcast in this low-interactivity scenario is significantly better; effectively double that of the high-interactivity workload, and more akin to the results obtained for the high-interactivity workload with small group sizes. This result is as expected, as reduced interactivity correspondingly results in fewer clients jumping between broadcast trees, fewer setup delays, and thus fewer untimely segments according to the clients' demands.

5.3.4.2 Additional Nodes

In all cases, as shown in Figure 5.11a and Figure 5.11b, the addition of extra resources in the form of additional content sources (*i.e.*, the roots of trees or seeds in swarms) is beneficial, with similar trends observed regardless of the delivery method being used. This is especially true of the case when content is made available in a redundant fashion rather than simply being 'striped' across nodes. The exact placement of these nodes also has some impact, albeit a less significant one, with a slightly diminishing effect as more nodes are made available.

5.3.4.3 Key Observations

From these findings, several points become clear. Regarding network costs in terms of traffic generated, pull P2P is relatively expensive in comparison with the simpler ALM approach. This cost is offset, however, by a resilience to large group sizes and high levels of interactivity in comparison with ALM. Fortunately, when the methods are combined into the hybrid method, network cost is lower than the pull peer-to-peer case, albeit higher than ALM alone, making it an effective compromise. In all cases, larger group sizes result in an increase in the amount of network traffic, and this also has a slightly negative effect on segment utility.

In terms of user satisfaction, larger groups have little effect on the fraction of timely segments received when the peer-to-peer system is involved under any level of interactivity. This is also largely true for the case of pure ALM when low levels of user interactivity are observed, but when users begin to jump around within a video, ALM performs significantly worse. This effect can be attributed to the lengthy join delays when the broadcast trees are particularly long, coupled with the standard waiting period on the broadcast channel to receive the segment(s) the user requires. Naturally, this does not affect the pull approach of the peer-to-peer method, and when this is combined with the ALM scheme, the peer-to-peer system acts as a means of patching; providing the media segments required in an on-demand fashion until the push scheme has stabilised. For similar reasons, segment utility is quite similar across all methods for low interactivity workloads, but when high levels of interactivity occur, there is a marked difference between the schemes. For instance, while the peer-to-peer approach typically achieves high-levels of user satisfaction, a larger percentage of segments are wasted relative to ALM. The naturally sequential nature of the broadcast over ALM may therefore be acting in its favour here, given that users are highly likely to want large numbers of media segments delivered in this fashion.

5.3.5 Conclusion

From the results observed, we can conclude that while peer-to-peer methodologies are certainly feasible for the delivery of interactive content from the user perspective, they are somewhat network-intensive. In contrast, a more traditional approach based around classical periodic broadcast techniques over an application-level multicast structure apparently work well for smaller numbers of passive viewers, but encounters problems when user interactivity and group size increase. The combination of these two approaches, broadly pull and push, can, however, offer a good compromise that provides adaptability to varying conditions in terms of audience size, interactivity levels, and the resources available. Across all the delivery methods considered, providing additional resources in terms of extra content nodes is beneficial, with their placement relative to clients being increasingly important dependent on their abundance.

It is therefore apparent that mixing a live distribution approach such as application-level multicast with an appropriate peer-to-peer patching mechanism over a typical network infrastructure (*i.e.*, typically lacking in IP multicast support) can provide a workable solution for delivery of on-demand video with interactivity support in a CDN environment. Given the wide variety of possible workloads, delivery methods and variables, much potential exists for future work in this field.

Chapter 6

Conclusion

This chapter is divided into three main sections, firstly, a brief overview of contributions this thesis has made. Followed by a discussion of possible future work, and finally wrapped-up with concluding remarks.

6.1 Thesis Contributions

This thesis has provided several contributions for the research community. For the first time, traces are now available from a highly interactive video-on-demand system, which show behaviour not previously observed. These traces have been analysed and modelled, to produce detailed and thorough user workloads, which will aid in the designing and testing of future video-on-demand techniques.

Additionally, the modelled behaviour exhibited usage patterns which were a complete departure from the classic start-to-finish models. When users are given a choice and the media is of an appropriate genre, users will only seek to and view the areas they are interested in. This is aided by the bookmark feature, which clearly influenced what was viewed. The combination of this produced *hotspots*, areas of high interest, which were common to all our videos.

With the models outlined in Chapter 4, it is clear that many existing delivery techniques do not perform well any more, as they were originally designed under the assumption of the start-to-finish model. However, with these new models it is possible to create new techniques which exploit these new usage patterns. Three new techniques were explained in Chapter 5. These include a method for detecting and moving ill-placed bookmarks, a method to pre-fetch far ahead of playback to popular segments, and finally, a new hybrid scheme which shows how peer-to-peer schemes operate under high interactivity.

The pre-fetching schemes builds up knowledge in real-time from the live system of what segments of the media are popular and in what order they are typically viewed. Using this knowledge it was demonstrated that users can successfully pre-fetch segments or *hotspots*, which they will probably view in the near future.

The hybrid peer-to-peer section outlined how existing peer-to-peer solutions were not appropriate for these interactive workload. This is supported by our experimentation which shows that a hybrid push/pull approach performed far better than existing peer-to-peer approaches.

6.2 Future Work

As bookmarks in our system were very popular, in a fully autonomic system the bookmarks should perhaps be created automatically. This could occur after the system has detected a large number of requests for a specific area of a video. A bookmark could then be provisionally placed and its position refined by a bookmark-moving algorithm.

Hotspots should also be detected automatically by the system, as their position and popularity can greatly be exploited. In this thesis we have only discussed the hotspot's length, but equally their start position can be inferred in the same (or similar) way to a bookmark's position. Alternative approaches could be taken which, for example, rank all the segments of the media by their popularity, and decide the top-N% should be hotspots. These rankings could easily be obtained using a cache replacement policy such as the least frequently used (LFU) or least recently used (LRU).

During our experiment, users were unhappy that we “spoilt the experience” of watching the sporting events covered somewhat, as the user could quickly determine the final outcome of the event from the bookmark names. The suggestion was made that we avoid labelling the bookmarks and instead simply describe them as points of interest. This could equally work if the bookmarks were autonomically created, since a system would be unable to name them itself. Note, unnamed bookmarks would only be useful if they are typically accessed sequentially, and not based on their name alone.

It was shown that pre-fetched bookmark hotspots only covered 35% of all viewed segments. Thus, pre-fetching schemes should consider more segments. This, of course, would make it harder to decide which segments to pre-fetch next. The cost of making a wrong decision could be reduced if the pre-fetching technique was modified, for example, pre-fetching more than one choice simultaneously.

Additionally, the data required for pre-fetching has not been fully exploited. There are numerous other uses for this data, such as producing management or business reports, better caching algorithms, construction of peer-to-peer overlays, or even deciding which segments to push to the user overnight. The cache algorithm could, for example, take into account any segments of the video which depend on another segment, and thus related segments are kept or evicted from the cache together. With peer-to-peer overlays, the network topology could fundamentally be structured based on the pre-fetching knowledge.

This thesis briefly looked at peer-to-peer delivery and it is clear that as demand for online media increases, techniques such as P2P will be used more. Many internet service providers (ISPs) are worried about P2P, as their networks were not designed or deployed to be symmetric. The typical home broadband connection has a higher downstream bitrate than upstream making it asymmetric. Yet, it has been shown that if the P2P topology is designed correctly, it can reduce the cost for the ISPs and content providers, with only a minor decrease in performance for the users [KRP05, HLR07].

This still needs further work to help balance the cost to ISPs, and the performance impact to the user, especially when considering the highly interactive workloads.

Many of the suggested solutions for delivering this content, such as peer-to-peer, are relatively simple for a standard desktop PC to use, however, more and more media is being consumed via mobile devices. 3G phones are being used to watch clips from the internet and Apple iPods are automatically downloading audio and video podcasts each day, providing their users with a "personal on demand broadcasting" service. These small mobile devices may not benefit from client-side improvements; instead the network should perhaps become more intelligent to be able to serve this new generation of device.

While not available today, future forms of interactivity may add an extra dimension to these problems. For example, systems which allow picture-in-picture (PiP), the ability to display one main video, with one or more smaller supplementary videos being displayed on top. Imagine a system where when watching a programme, areas of the video could be highlighted to form a hyperlink to more information. This hyperlinked information would then be displayed with picture-in-picture technology. This can be considered useful in many situations, for example, finding statistics about players in sporting events, viewing more information on a news article, or reading reviews or production notes about a film currently being shown. Now, videos could hyperlink between each other, not just causing links between *hotspots*, but now between videos.

While not the case for all content, high levels of interactivity are becoming more common, whilst users are both relying on and expecting video-on-demand services to provide more advanced interactive functionality. Our study suggests that CDN mechanisms must improve to handle more diverse applications, content and users. To achieve this, the development of new algorithms must be driven by models derived from realistic characterised workloads. The development of such strategies is reliant on gaining a deeper understanding of the relevant workload parameters. The analysis and models presented in this thesis aim to aid in this endeavour.

6.3 Conclusion

We have presented a study and characterisation of user behaviour for our interactive Video-on-Demand system. We note that by adding simple bookmarks to points of interest within the media, the access patterns are greatly influenced. This behaviour led to high levels of seeking which created relatively short and sparsely distributed segments whose popularity was orders of magnitude more popular than other segments.

Many existing delivery mechanisms are not designed for high levels of interactive behaviour and are instead optimised for classic start-to-finish streaming. Content distribution techniques must therefore adapt to efficiently handle these kinds of access patterns. They could, for example, take advantage of the power-law distributions of segment popularity by replicating those that generate the most demand. For instance, we observed that 10% of segments accounted for 44% of all requests.

The departure from classic start-to-finish playback encourages the design of agile delivery mechanisms that allow quick seeking, and expect certain segments to be more popular. We have seen that adding bookmarks will highly influence the order in which users view the content, making the sequence of actions somewhat predictable. This can then be exploited by allowing users to pre-fetch content that they are predicted to need shortly, thus reducing any delays that they are likely to experience. However, we noted that bookmarks could be harmful by causing unnecessary seeks if incorrectly placed. This could be remedied for both client and server by simply moving the bookmark autonomically based on observed user behaviour.

Advances in pull-based peer-to-peer VoD can aid in agile delivery, however the overheads associated make it unacceptable in some situations. Instead, a combination of push-based peer-to-peer delivery, which typically does not handle seeking well, and pull-based, can produce an efficient delivery platform for these interactive workloads.

So far we have only considered bookmarks within music and sport videos, but bookmarks are equally applicable in many other genres. For example, bookmarks are commonly found in the form of chapters on video DVDs. It is not clear if the same high levels of interactivity would be observed in such media, or if the classic start-to-finish model would still be prevalent.

In conclusion, we are entering a new era of video-on-demand, one where media is being consumed in abundance, on a myriad of devices. Our VoD systems must be flexible and agile to support current and future trends, as well as to take advantage of new techniques such as peer-to-peer, or to expect new user behaviours such as those demonstrated in this thesis. Overall, this is an exciting new future for online media, and one which provides many opportunities for improvement.

Bibliography

[408] Channel 4. 4oD, 2008. `http://www.channel4.com/4od/`. [Cited on page 9]

[AGRM06] S. Annapureddy, C. Gkantsidis, P. Rodriguez, and L. Massoulie. Providing video-on-demand using peer-to-peer networks. In *Proceedings of Internet Protocol TeleVision (IPTV) Workshop*, volume 6, 2006. [Cited on page 17]

[AKEV01] J.M. Almeida, J. Krueger, D.L. Eager, and M.K. Vernon. Analysis of educational media server workloads. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, pages 21–30. ACM Press New York, NY, USA, 2001. [Cited on pages 31, 33, 34, 57, and 62]

[AN07] M. Ames and M. Naaman. Why we tag: motivations for annotation in mobile and online media. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 971–980. ACM Press New York, NY, USA, 2007. [Cited on page 7]

[And06] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, July 2006. [Cited on pages 28 and 30]

[ASP00] S. Acharya, B. Smith, and P. Parnes. Characterizing user access to videos on the world wide web. In *Proceedings of the ACM/SPIE Multimedia Computing and Networking (MMCN)*, January 2000. [Cited on page 27]

[AWY96] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A permutation-basaed pyramid broadcasting scheme for video-on-demand. In *Proceedings of the IEEE Int☒l Conf. on Multimedia Systems*, June 1996. [Cited on page 84]

[BAE03] AC Begen, Y. Altunbasak, and O. Ergun. Multi-path selection for multiple description encoded video streaming. In *Proceedings of IEEE International Conference on Communications (ICC)*, volume 3, 2003. [Cited on page 16]

[BBC08] BBC. iPlayer, 2008. `http://www.bbc.co.uk/iplayer/`. [Cited on pages 5, 9, and 26]

[BBK02] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable application layer multicast. *SIGCOMM Computer Communication Review*, 32(4):205–217, 2002. [Cited on page 16]

[BHH^{+}94] R.O. Banker, J.B. Huppertz, M.T. Hayashi, D.B. Lett, V.E. Godlewski, and M.W. Raley. Method of providing video on demand with vcr like functions, oct 1994. US Patent 5,357,276. [Cited on page 85]

[BHP05] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. Analyzing and improving bittorrent performance. Technical Report MSR-TR-2005-03, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, February 2005. [Cited on page 21]

[BL99] S. Bakiras and V.O.K Li. Smoothing and prefetching video from distributed servers. *Proceedings of the International Conference on Network Protocols (ICNP)*, pages 311–318, 1999. [Cited on page 76]

[BNLT08] A. Bar-Noy, R.E. Ladner, and T. Tamir. Optimal delay for media-on-demand with pre-loading and pre-buffering. *Theoretical Computer Science*, 2008. [Cited on page 75]

[BP05] Patrick Barwise and Sarah Pearson. Fast-forward puts TV advertising to the test. Financial Times, Oct 2005. [Cited on page 37]

[BSk08] BSkyB. Sky player, 2008. `http://sky.com/skyplayer/`. [Cited on page 9]

[BWS^{+}01] M.K. Bradshaw, B. Wang, S. Sen, L. Gao, J. Kurose, P. Shenoy, and D. Towsley. Periodic broadcast and patching services implementation, measurement, and analysis in an internet streaming video testbed. In *Proceedings of ACM Multimedia*, 2001. [Cited on page 86]

[CCB^{+}04] C.P. Costa, I.S. Cunha, A. Borges, C.V. Ramos, M.M. Rocha, J.M. Almeida, and B. Ribeiro-Neto. Analyzing client interactivity in streaming media. In *Proceedings of the 13th international conference on World Wide Web*, pages 534–543. ACM New York, NY, USA, 2004. [Cited on pages 32, 33, 34, 53, 54, and 58]

[CCL08] Y. Chen, C. Chen, and C. Li. A measurement study of cache rejection in P2P live streaming system. *28th International Conference on Distributed Computing Systems Workshops (ICDCS)*, pages 12–17, 2008. [Cited on page 19]

[CDK^{+}03] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, pages 298–313, New York, NY, USA, 2003. ACM. [Cited on page 16]

[CDL07] X. Cheng, C. Dale, and J. Liu. Understanding the characteristics of internet short video sharing: YouTube as a case study. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, 2007. [Cited on pages 7 and 8]

[CG02] Ludmila Cherkasova and Minaxi Gupta. Characterizing locality, evolution, and life span of accesses in enterprise media server workloads. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, pages 33–42, New York, NY, USA, 2002. ACM. [Cited on page 27]

[CG04] Ludmila Cherkasova and Minaxi Gupta. Analysis of enterprise media server workloads: access patterns, locality, content evolution, and rates of change. *IEEE/ACM Transactions on Networking (TON)*, 12(5):781–794, 2004. [Cited on pages 29 and 60]

[CH99] Y. Cai and K.A. Hua. An efficient bandwidth-sharing technique for true video on demand systems. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 211–214. ACM Press New York, NY, USA, 1999. [Cited on page 86]

[Chi95] T. Chiueh. A periodic broadcasting architecture for large-scale residential video-on-demand service. In *Proceedings of SPIE*, volume 2615, pages 162–169, 1995. [Cited on page 84]

[CHV99] Ying Cai, K.A. Hua, and Khanh Vu. Optimizing patching performance. In *Proceedings of SPIE Conference on Multimedia Computing and Networking (MMCN)*, volume 3654, pages 204–215, 1999. [Cited on pages 85 and 86]

[CKR$^+$07] M. Cha, H. Kwak, P. Rodriguez, Y.Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM New York, NY, USA, 2007. [Cited on pages 28, 29, and 50]

[CL97] Steven W. Carter and Darrell D.E. Long. Stream tapping: A system for improving efficiency on a video-on-demand server. Technical report, University of California at Santa Cruz, Santa Cruz, CA, USA, 1997. [Cited on page 85]

[Cli08] Clip2. The gnutella protocol specification v0.4, 2008. `http://www.clip2.com`. [Cited on page 15]

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition, 2001. [Cited on page 19]

[Coh03] Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, California, USA, June 2003. [Cited on pages 15, 17, and 20]

[Com00] Avanstar Communications. Video store magazine, March 2000. `http://www.videostoremag.com/`. [Cited on page 28]

[CRSZ02] Y. Chu, SG Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471, 2002. [Cited on page 15]

[CSWZ03] S. Chen, B. Shen, S. Wee, and X. Zhang. Adaptive and lazy segmentation based proxy caching for streaming media delivery. In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 22–31, 2003. [Cited on page 59]

[CT03] Y. Chung and AH Tewfik. An efficient video broadcasting protocol with scalable preloading scheme. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, volume 1, 2003. [Cited on pages 75 and 77]

[CWVL01] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming media workload. In *Proceedings of the 2001 USENIX Symp. on Internet Technologies and Systems*, 2001. [Cited on pages 25, 26, 31, 36, and 57]

[CY97] K. Chinen and S. Yamaguchi. An interactive prefetching proxy server for improvement of www latency. In *Proceedings of INET*, volume 97, 1997. [Cited on page 76]

[CZ] K. Calvert and E. Zegura. Georgiatech internetwork topology models. http://www.cc.gatech.edu/projects/gtitm. [Cited on page 90]

[Dai08] Dailymotion, 2008. `www.dailymotion.com`. [Cited on page 6]

[Dat03] The Internet Movie Database, August 2003. `http://www.imdb.com/`. [Cited on page 28]

[Dav01] B.D. Davison. Assertion: Prefetching with GET is not good. In *Proceedings of the Sixth International Workshop Web Caching and Content Distribution*, pages 203–215, 2001. [Cited on page 76]

[DHRS07] P. Dhungel, X. Hei, K.W. Ross, and N. Saxena. The pollution attack in P2P live video streaming: measurement results and defenses. In *Proceedings of the workshop on Peer-to-Peer streaming and IP-TV*, pages 323–328. ACM New York, NY, USA, 2007. [Cited on page 16]

[DHT04] TT Do, KA Hua, and MA Tantaoui. P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment. In *Proceedings of the IEEE International Conference on Communications*, volume 3, 2004. [Cited on page 16]

[DSS94] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proceedings of the second ACM international conference on Multimedia*, pages 15–23, October 1994. [Cited on pages 26 and 84]

[DSST95] Asit Dau, Perwez Shahabuddin, Dinkar Sitaram, and Don Towsley. Channel allocation under batching and VCR control in video-on-demand systems. *Journal of Parallel and Distributed Computing*, 30(2):168–179, November 1995. [Cited on page 87]

[Duc99] D. Duchamp. Prefetching hyperlinks. In *Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems*, volume 2, pages 12–12. USENIX Association Berkeley, CA, USA, 1999. [Cited on page 76]

[EFV99] D.L. Eager, M.C. Ferris, and M.K. Vernon. Optimized regional caching for on-demand data delivery. In *Proceedings of SPIE Multimedia Computing and Networking (MMCN)*, volume 3654, pages 301–316, 1999. [Cited on page 76]

[eMu08] eMule, 2008. `http://www.emule.com/`. [Cited on page 15]

[EVZ99] D. Eager, M. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for video-on-demand servers. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 199–202. ACM Press New York, NY, USA, 1999. [Cited on page 86]

[EVZ01] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):742–757, Sep 2001. [Cited on page 86]

[FA06] G. S. Fishman and I. J. B. F. Adan. How heavy-tailed distributions affect simulation-generated time averages. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 16(2):152–173, 2006. [Cited on page 58]

[Fac08] Facebook. flvtool++, 2008. `http://mirror.facebook.com/facebook/flvtool++/`. [Cited on page 44]

[FFm08] FFmpeg, 2008. `http://ffmpeg.mplayerhq.hu/`. [Cited on pages 40 and 42]

[FGM$^+$99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC2616 - Hypertext transfer protocol – HTTP/1.1, June 1999. [Cited on page 45]

[FM05] Geoffrey A. Fowler and Sarah McBride. Newest export from china: Pirated pay tv. Wall Street Journal, September 2005. [Cited on page 18]

[Fro06] Dan Frommer. Your tube, whose dime?, April 2006. `http://www.forbes.com/intelligentinfrastructure/2006/04/27/video-youtube-myspace_cx_df_0428video.html`. [Cited on page 23]

[GA04] M. Guo and MH Ammar. Scalable live video streaming to cooperative clients using time shifting and video patching. In *Proceedings of the 23rd*

Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), volume 3, 2004. [Cited on page 88]

[GALM07] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube traffic characterization: a view from the edge. *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28, 2007. [Cited on page 7]

[GBW97] Carsten Griwodz, Michael Bär, and Lars C. Wolf. Long-term movie popularity models in video-on-demand systems: or the life of an on-demand movie. In *Proceedings of the fifth ACM international conference on Multimedia*, pages 349–357, New York, NY, USA, 1997. ACM. [Cited on page 27]

[GCXZ05] L. Guo, S. Chen, Z. Xiao, and X. Zhang. Analysis of multimedia workloads with implications for internet streaming. In *Proceedings of the 14th international conference on World Wide Web*, pages 519–528. ACM Press New York, NY, USA, 2005. [Cited on pages 7, 31, and 32]

[GDS+03] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 314–329. ACM Press New York, NY, USA, 2003. [Cited on page 27]

[GLZS00] C. Griwodz, M. Liepert, M. Zink, and R. Steinmetz. Tune to lambda patching. *ACM SIGMETRICS Performance Evaluation Review*, 27(4):20–26, 2000. [Cited on page 86]

[Gri08] GridCast, 2008. `http://www.gridcast.cn/`. [Cited on page 20]

[GT99] L. Gao and D. Towsley. Supplying instantaneous video-on-demand services using controlled multicast. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, volume 2, 1999. [Cited on page 86]

[HCS98] Kien A. Hua, Ying Cai, and Simon Sheu. Patching: A multicast technique for true video-on-demand services. In *Proceedings of the sixth ACM international Conference on Multimedia*, pages 191–200, New York, NY, USA, 1998. ACM. [Cited on pages 85 and 86]

[HFC+08] Y. Huang, T.Z.J. Fu, D.M. Chiu, J.C.S. Lui, and C. Huang. Challenges, design and analysis of a large-scale P2P-VoD system. In *Proceedings of ACM SIGCOMM*. ACM New York, NY, USA, 2008. [Cited on page 19]

[HLR07] Cheng Huang, Jin Li, and Keith W. Ross. Can internet video-on-demand be profitable? *SIGCOMM Computer Communication Review*, 37(4):133–144, 2007. [Cited on pages 7, 32, 34, 77, and 100]

[HLR08] Xiaojun Hei, Yong Liu, and K.W. Ross. IPTV over P2P streaming networks: The mesh-pull approach. *IEEE Communications Magazine*, 46(2):86–92, February 2008. [Cited on page 17]

[HNG$^+$99] M. Hofmann, T.S.E. Ng, K. Guo, S. Paul, and H. Zhang. Caching techniques for streaming multimedia over the internet. *in Bell Labs Technical Memorandum*, 1999. [Cited on page 38]

[HS97] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proceedings of ACM SIGCOMM*, September 1997. [Cited on page 84]

[Hua07] Gale Huang. Experiences with pplive. Peer-to-Peer Streaming and IP-TV Workshop (P2P-TV), August 2007. Keynote Presentation. [Cited on page 19]

[Hul08] Hulu, 2008. `http://www.hulu.com/`. [Cited on page 6]

[HWLR08] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Understanding hybrid CDN-P2P: Why limelight needs its own red swoosh. In *Proceedings of the 18th International workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 75–80, May 2008. [Cited on page 9]

[Inc08a] DirectTV Inc. ReplayTV, 2008. `http://www.replay.com/`. [Cited on page 12]

[Inc08b] Kontiki Inc. Kontiki, 2008. `http://www.kontiki.com/`. [Cited on page 10]

[Joo08] Joost, 2008. `http://www.joost.com`. [Cited on pages 20 and 21]

[JT97] Li-Shen Juhn and Li-Ming Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3):268–271, September 1997. [Cited on page 85]

[KPS08] Santosh Kulkarni, Jehan-François Pâris, and Purvi Shah. A stream tapping protocol involving clients in the distribution of videos on demand. *Adv. MultiMedia*, 8(2):1–9, 2008. [Cited on page 85]

[KRAV03] Dejan Kostić, Adolfo Rodriguez, Jeannie Albrecht, and Amin Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP)*, pages 282–297, New York, NY, USA, 2003. ACM. [Cited on page 17]

[KRP05] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *Proceedings of the Internet Measurement Conference (IMC)*, pages 63–76. USENIX Association Berkeley, CA, USA, 2005. [Cited on page 100]

[KS08] U.R. Krieger and R. Schwessinger. Analysis and quality assessment of peer-to-peer IPTV systems. In *Proceedings of the IEEE International Symposium on Consumer Electronics (ISCE)*, pages 1–4, 2008. [Cited on pages 19 and 22]

[LCKN05] Mingzhe Li, Mark Claypool, Robert Kinicki, and James Nichols. Characteristics of streaming media stored on the web. *ACM Transactions on Internet Technology (TOIT)*, 5(4):601–626, 2005. [Cited on page 26]

[LGL08] Y. Liu, Y. Guo, and C. Liang. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, 2008. [Cited on page 15]

[LJL$^+$06] X. Liao, H. Jin, Y. Liu, L.M. Ni, and D. Deng. AnySee: Peer-to-peer live streaming. In *Proceedings of IEEE INFOCOM*, pages 1–10, 2006. [Cited on page 17]

[LNZ07] J. Li, K. Nahrstedt, and H. Zhang. Editorial special section on peer-to-peer video streaming. *Multimedia, IEEE Transactions on*, 9(8):1551–1553, 2007. [Cited on page 15]

[Low08] Josh Lowensohn. Dailymotion gets high-definition videos. Cnet news, February 2008. [Cited on page 8]

[LSA01] E. Limpert, W.A. Stahel, and M. Abbt. Log-normal distributions across the sciences: keys and clues. *Bioscience*, 51(5):341–352, 2001. [Cited on page 50]

[Ltd08] BSkyB Ltd. Sky+, 2008. `http://www.sky.com/portal/site/skycom/skyproducts/skytv/skyplus`. [Cited on page 11]

[MCH01] Laurent Mathy, Roberto Canonico, and David Hutchison. An overlay tree building control protocol. In *Proceedings of the 3rd International Workshop on Networked Group Communication (NGC)*, November 2001. [Cited on pages 16 and 87]

[Med08] Virgin Media. On Demand, 2008. `http://www.virginmedia.com/tvradio/ondemand/`. [Cited on page 13]

[Met08] Metacafe, 2008. `http://www.metacafe.com`. [Cited on page 6]

[Mic08a] Microsoft. Mediaroom, 2008. `http://www.microsoftmediaroom.com/`. [Cited on page 14]

[Mic08b] Microsoft. UltimateTV, 2008. `http://www.ultimatetv.com/`. [Cited on page 12]

[Mil07] N. Miller. Manifesto for a new age. *Wired Magazine*, March 2007. [Cited on page 6]

[Mit04] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251, 2004. [Cited on page 58]

[MR07] N. Magharei and R. Rejaie. PRIME: Peer-to-peer Receiver-drIven MEsh-based streaming. In *Proceedings of IEEE INFOCOM*, 2007. [Cited on page 17]

[MSW05] H. Ma, G.K. Shin, and W. Wu. Best-effort patching for multicast true vod service. *Multimedia Tools and Applications*, 26(1):101–122, 2005. [Cited on page 86]

[MV05] Laurent Massoulié and Milan VojnoviĆ. Coupon replication systems. In *Proceedings of the 2005 ACM International conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 2–13, New York, NY, USA, 2005. ACM. [Cited on page 20]

[MyS08] MySpaceTV, 2008. `http://vids.myspace.com/`. [Cited on page 6]

[Nap08] Napster, 2008. `http://www.napster.com`. [Cited on page 15]

[Net02] Sharman Networks. Kazaa, January 2002. `http://www.kazaa.com/`. [Cited on pages 21 and 27]

[Onl08a] BBC News Online. BBC announces Nintendo Wii deal, April 2008. `http://news.bbc.co.uk/1/hi/technology/7338344.stm`. [Cited on page 11]

[Onl08b] BBC News Online. BBC iPlayer comes to the iPhone, March 2008. `http://news.bbc.co.uk/1/hi/technology/7283702.stm`. [Cited on page 11]

[OP07] Martin Olausson and Jim Penhune. Global IPTV forecast: Homes, users and subscribers. Technical report, Strategy Analytics, January 2007. [Cited on page 13]

[Pâr01] J.F. Pâris. A stream tapping protocol with partial preloading. In *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 423–431. IEEE Computer Society Washington, DC, USA, 2001. [Cited on page 75]

[Pâr02] J.F. Pâris. A broadcasting protocol for video-on-demand using optional partial preloading. In *Proceedings of the 11th International Conference on Computing, I*, page 319, 2002. [Cited on page 75]

[PFS08] PFSVOD, 2008. `http://www.pplive.com/subject/20070808pfsvod/`. [Cited on page 20]

[PGM$^+$06] X.G. Pañeda, R. Garcia, D. Melendi, M. Vilas, and V. Garcia. Popularity analysis of a video-on-demand service with a great variety of content types. influence of the subject and video characteristics. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA)*, pages 29–34. IEEE Computer Society Washington, DC, USA, 2006. [Cited on pages 29 and 30]

[PK98] J. Padhye and J. Kurose. An empirical study of client interactions with a continuous-media courseware server. In *Proceedings of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 1998. [Cited on page 33]

[PK99] Jitendra Padhye and Jim Kurose. Continuous-media courseware server: A study of client interactions. *IEEE Internet Computing*, 03(2):65–73, 1999. [Cited on pages 33 and 57]

[PL01] J. Pâris and DDE Long. The case for aggressive partial preloading in broadcasting protocols for video-on-demand. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 112–115, 2001. [Cited on page 75]

[Plc08] YouGov Plc. Redback networks' on-demand TV and internet video survey, May 2008. [Cited on pages 11 and 25]

[PLM99] J.F. Pâris, D.D.E. Long, and P.E. Mantey. Zero-delay broadcasting protocols for video-on-demand. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 189–197. ACM New York, NY, USA, 1999. [Cited on page 75]

[PPL08] PPLive, 2008. `http://www.pplive.com`. [Cited on pages 18 and 19]

[PPS08] PPStream, 2008. `http://www.ppstream.com/`. [Cited on pages 18 and 20]

[Pur08] Rob Purchese. Fan makes BBC iPlayer work on PS3, April 2008. `http://www.eurogamer.net/article.php?article_id=132029`. [Cited on page 11]

[RMC$^+$05] M. Rocha, M. Maia, Í. Cunha, J. Almeida, and S. Campos. Scalable media streaming to interactive users. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 966–975. ACM Press New York, NY, USA, 2005. [Cited on page 86]

[Roo06] Jelle Roozenburg. Secure decentralized swarm discovery in tribler. Master's thesis, Parallel and Distributed Systems Group, Delft University of Technology, November 2006. [Cited on page 20]

[RR97] M. Reisslein and K.W. Ross. A join–the–shortest–queue prefetching protocol for vbr video on demand. *Proceedings of the International Conference on Network Protocols (ICNP)*, 1997. [Cited on page 76]

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. [Cited on page 10]

[SF07] T. Silverston and O. Fourmaux. Measuring P2P IPTV systems. In *Proceedings of the 17th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, 2007. [Cited on pages 19 and 46]

[SGRT99] S. Sen, L. Gao, J. Rexford, and D. Towsley. Optimal patching schemes for efficient multimedia streaming. In *Proceedings of the 9th International*

Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 1999. [Cited on page 86]

[Sky08] Skype, 2008. `http://www.skype.com`. [Cited on page 21]

[SLP^{+}06] Y. Shen, Z. Liu, S. Panwar, K. Ross, and Y. Wang. On the design of prefetching strategies in a peer-driven video on-demand system. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 817–820, 2006. [Cited on page 77]

[SMZ04] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the internet. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54. ACM Press New York, NY, USA, 2004. [Cited on pages 34, 35, and 36]

[SOP08] SOPCast, 2008. `http://www.sopcast.com/`. [Cited on page 18]

[SP07a] P. Shah and J.-F. Pâris. Peer-to-peer multimedia streaming using BitTorrent. In *Proceedings of the 26th International Performance of Computers and Communication Conference (IPCCC)*, New Orleans, Louisiana, USA, April 2007. [Cited on page 88]

[SP07b] P. Shah and J.F. Pâris. Peer-to-peer multimedia streaming using BitTorrent. In *Proceedings of the 26th IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 340–347, 2007. [Cited on page 20]

[SRT99] Subhabrata Sen, Jennifer Rexford, and Donald F. Towsley. Proxy prefix caching for multimedia streams. In *Proceedings of IEEE INFOCOM*, pages 1310–1319, April 1999. [Cited on pages 32, 38, and 76]

[SSF08] Mohit Saxena, Umang Sharan, and Sonia Fahmy. Analyzing video services in web 2.0: A global perspective. In *Proceedings of the 18th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2008. [Cited on pages 7, 9, and 26]

[Tel08] British Telecom. BT Vision, 2008. `http://www.btvision.bt.com/`. [Cited on page 13]

[TEVG02] H. Tan, D.L. Eager, M.K. Vernon, and H. Guo. Quality of service evaluations of multicast streaming protocols. In *Proceedings of ACM SIGMETRICS*, June 2002. [Cited on page 86]

[THD03] DA Tran, KA Hua, and T. Do. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proceedings of IEEE INFOCOM*, pages 1283–1292, 2003. [Cited on page 16]

[TT08] TiVo Technologies. TiVo, 2008. `http://www.tivo.com/`. [Cited on page 12]

[TVA08] TVAnts, 2008. `http://www.tvants.com/`. [Cited on page 18]

[UUS08] UUSee, 2008. `http://www.uusee.com`. [Cited on page 20]

[VAM$^+$02] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin. A hierarchical characterization of a live streaming media workload. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 117–130. ACM Press New York, NY, USA, 2002. [Cited on page 25]

[VAMJ$^+$06] E. Veloso, V. Almeida, W. Meira Jr, A. Bestavros, and S. Jin. A hierarchical characterization of a live streaming media workload. *IEEE/ACM Transactions on Networking (TON)*, 14(1):133–146, 2006. [Cited on page 35]

[VdMSK02] J. Van der Merwe, S. Sen, and C. Kalmanek. Streaming video traffic: Characterization and network impact. In *Proceedings of the Seventh International Web Content Caching and Distribution Workshop (WCW)*, 2002. [Cited on page 35]

[VGLN07] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt. Measurement of a large-scale overlay for multimedia streaming. In *Proceedings of the 16th international symposium on High performance distributed computing (HPDC)*, pages 241–242. ACM Press New York, NY, USA, 2007. [Cited on page 19]

[VI96] S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia Systems*, 4(4):179–208, August 1996. [Cited on page 84]

[Vid08a] Google Video, 2008. `http://video.google.com/`. [Cited on page 6]

[Vid08b] MSN Video, 2008. `http://video.msn.com/`. [Cited on page 6]

[Vid08c] VideoLan. Vlc media player, 2008. `http://www.videolan.org/`. [Cited on page 42]

[VIF06a] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiToS: Enhancing BitTorrent for supporting streaming applications. *IEEE Global Internet*, 2006. [Cited on page 20]

[VIF06b] Aggelos Vlavianos, Marios Iliofotou, and Michalis Faloutsos. BiToS; enhancing BitTorrent for supporting streaming applications. In *Proceedings of the 9th IEEE Global Internet Symposium (GI)*, Barcelona, Spain, April 2006. [Cited on page 88]

[VPG$^+$05] M. Vilas, XG Paneda, R. Garcia, D. Melendi, and VG Garcia. User behaviour analysis of a video-on-demand service with a widevariety of subjects and lengths. In *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 330–337, 2005. [Cited on pages 33, 54, and 62]

[Vuz08] Vuze, 2008. `http://www.vuze.com/`. [Cited on page 20]

[Wik08] Wikipedia. Article on Adobe Flash, 2008. `http://en.wikipedia.org/wiki/Adobe_Flash`. [Cited on page 8]

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998. [Cited on page 19]

[XLKZ07] Susu Xie, Bo Li, G.Y. Keung, and Xinyan Zhang. Coolstreaming: Design, theory, and practice. *IEEE Transactions on Multimedia*, 9(8):1661–1671, December 2007. [Cited on page 18]

[YLE04] CK Yeo, BS Lee, and MH Er. A survey of application level multicast techniques. *Computer Communications*, 27(15):1547–1568, 2004. [Cited on page 15]

[You08] YouTube, 2008. `http://youtube.com/`. [Cited on pages 5, 6, and 25]

[YZZZ06] H. Yu, D. Zheng, B.Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proceedings of the 2006 EuroSys conference*, pages 333–344. ACM Press New York, NY, USA, 2006. [Cited on pages 28, 30, 31, 32, and 57]

[Zat08] Zattoo, 2008. `http://www.zattoo.com`. [Cited on page 18]

[Zip49] G. K. Zipf. Human behavior and the principal of least-effort, 1949. [Cited on page 26]

[ZLLY05] X. Zhang, J. Liu, B. Li, and T.S.P. Yum. Coolstreaming/DONet: A data-driven overlay network for efficient live media streaming. In *Proceedings of IEEE INFOCOM*, volume 3, pages 13–17, 2005. [Cited on page 18]