TECHNISCHE
UNIVERSITÄT
DARMSTADT

PEER-ASSISTED VIDEO-ON-DEMAND: COST REDUCTION AND PERFORMANCE ENHANCEMENT FOR USERS, OVERLAY PROVIDERS, AND NETWORK OPERATORS

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
in englischer Sprache genehmigte Dissertation

von

DIPL.-INFORM. KONSTANTIN PUSSEP

Geboren in Nowosibirsk, Russland

Vorsitz: Prof. Dr.-Ing. Gerd Balzer
Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Phuoc Tran-Gia

Tag der Einreichung: 29. November 2010
Tag der Disputation: 11. Februar 2011

Hochschulkennziffer D17
Darmstadt 2011

ABSTRACT

Peer-assisted content delivery is an attractive way to distribute video content through the Internet at low costs. This approach combines the scalability of the peer-to-peer paradigm, in which users contribute their resources, and the service level guarantees of server-based systems. Thus, peer-assistance enables a content provider to reduce its server hosting costs, which is crucial in a commercial scenario. However, to be successful, such systems must take into account the interests of all three stakeholders involved: (1) *users* that demand high streaming quality with low fees and limited resource contribution, (2) *content providers* that aim to decrease server hosting costs, and (3) *network operators* that aim to avoid inefficient use of their infrastructure due to the network-oblivious behavior of peer-assisted overlays.

In this thesis, we address these requirements and develop adaptive mechanisms to achieve a benefit for all three stakeholders, resulting in the so-called *triple-win situation*. Our main scenario is video-on-demand streaming, in which users can request pre-stored video content at any time and watch the video while downloading it. Thereby, video-on-demand streaming imposes stricter requirements compared to other systems that utilize peer resources, such as BitTorrent-like file sharing. Ideally, the video playback should start within few seconds and there should be no playback stalling.

First, we focus on dedicated servers or caches that are essential resources in peer-assisted streaming systems. Their provision is necessary to guarantee a satisfying quality of experience to consumers, yet they cause significant and largely avoidable costs for the content provider, which can be minimized. The high dynamics of uncontrolled peers, however, result in unpredictable changes of the resource demand. Since peers additionally offer services to other peers, the supply of resources is also dynamic. This behavior makes the management of peer-assisted systems and the proper allocation of resources challenging. This thesis proposes *adaptive allocation policies*, a new approach to address this issue. The policies estimate the capacity situation and service demand of the system to adaptively optimize allocated resources. Extensive simulations, verified by testbed measurements, prove the efficiency of our approach, which achieves a more competitive performance than well-dimensioned static systems.

In the next step, we examine content delivery overlays from the network operators' perspective, since such overlays are responsible for a large amount of consumer traffic, including the costly inter-domain traffic. The existing approaches often fail to satisfy the requirements of all involved stakeholders. We propose a novel *incentive-based traffic management mechanism* where a network operator offers additional free resources to selected users. The mechanism assigns resources to users that behave compliant with the network and overlay policies. Our evaluation shows that this approach satisfies the requirements of network operators *and* overlay participants (provider and users). To this end, the proposed mechanism is able to reduce the inter-domain traffic while improving the overlay performance. We also show that even a single network operator can successfully apply the proposed mechanism.

Finally, we consider the availability of peer resources in peer-assisted content distribution. Besides contributing upload bandwidth, it is important that peers stay online after finishing their downloads to serve new download requests. The longer a peer stays online the more it can help to offload the servers. However, too extensive online time often results in high energy consumption paid by users without an adequate benefit to the system. In upcoming decentralized architectures based on *set-top boxes* that act as tiny servers, their energy consumption can even dominate distribution costs. Therefore, we propose advanced *standby policies* that reduce the energy consumption

of set-top boxes while still offloading servers significantly. We evaluate the standby polices in a lifelike scenario. The results show that a near-optimal behavior can be realized by utilizing common features of set-top boxes such as the wake-up timer. We further extend a standby policy with network awareness to address the needs of network operators. In this regard, the resulting policy takes into account the interests of all three stakeholders: users, content providers, and network operators.

KURZFASSUNG

Peer-unterstützte Inhaltsverteilsysteme sind eine vielversprechende Alternative, um Videoinhalte über das Internet auszuliefern. Dieser Ansatz kombiniert die Skalierbarkeit und Kosteneffizienz des Peer-to-Peer *(P2P)*-Paradigmas, wobei Nutzer ihre Ressourcen dem System zur Verfügung stellen, mit den Dienstgütegarantien der Client-Server-Systeme. Insbesondere in kommerziellen Szenarien führt dies zu entscheidenden Vorteilen. Zum einen können durch die Server die Qualitätsanforderungen der Nutzer befriedigt werden. Zum anderen können durch die Verwendung der Nutzerressourcen die Serverkosten für den Inhaltsanbieter reduziert werden.

Für den Erfolg eines Peer-unterstützten Systems müssen allerdings die Interessen von allen drei involvierten Parteien beachtet werden: (1) *Nutzer* erwarten hohe Wiedergabequalität bei geringen Kosten (Downloadgebühren und beigetragene Ressourcen); (2) *Inhaltsanbieter* wollen ihre Distributionskosten gering halten und gleichzeitig viele Nutzer unterstützen; (3) *Netzbetreiber* befürchten übermäßige Belastung ihrer Netzwerkinfrastruktur wegen der hohen Bandbreitenanforderungen und der häufigen Missachtung der Netzwerktopologie bei Videoverteilung.

Diese Dissertation adressiert diese Anforderungen und entwickelt adaptive Mechanismen mit dem Ziel, eine zufriedenstellende Situation für die beteiligten Parteien zu erreichen. Das Hauptszenario ist dabei das sogenannte *Video-on-Demand-Streaming*, bei dem Nutzer jederzeit digitale Videoinhalte über das Internet herunterladen und währenddessen wiedergeben können. Dabei werden die speziellen Anforderungen des Video-on-Demand berücksichtigt, denn Videowiedergabe sollte nach nur nach wenigen Sekunden starten und möglichst ohne Unterbrechungen wiedergegeben werden.

Als erstes untersucht diese Dissertation wie dedizierte Server (oder Caches im Allgemeinen) effizient eingesetzt werden können, um die Streaming-spezifische Dienstgüte zu garantieren. Wegen der damit verbundenen Kosten für Bereitstellung und Datenauslieferung muss der Ressourcenverbrauch der Inhaltsanbieter und Nutzer minimiert werden. Dies wird insbesondere durch das dynamische und oft unvorhersehbare Verhalten der Nutzer zur Herausforderung, da die Nutzer nicht nur als Abnehmer sondern auch als Anbieter von Daten agieren. Die daraus resultierenden Nachfrage- und Angebotsschwankungen für einzelne Videos (und Videoteile) erschweren eine effiziente Bereitstellung der Ressourcen verglichen mit einem reinen Client-Server-System. Als Lösung werden in dieser Arbeit *adaptive Bereitstellungsmechanismen* vorgestellt, welche die verfügbaren Server (oder Caches) effizient einsetzen, um Nachfrage und Angebot im System im Gleichgewicht zu halten. Die Qualität der entwickelten Mechanismen wird mittels ausführlicher Simulationen bewertet und durch Messungen in einem Testbed verifiziert. Dabei zeigt es sich, dass die vorgeschlagenen Mechanismen, im Vergleich zu gut-dimensionierten statischen Systemen, die erforderliche Dienstgüte bei niedrigeren Kosten erreichen.

Im nächsten Schritt werden die Auswirkungen der Inhaltsverteilsysteme auf die Netzbetreiber betrachtet. Der Grund dafür ist, dass Videoinhalte den immer größeren Anteil des Datenverkehrs im Internet ausmachen und eine entsprechend hohe Last auf der Netzwerkinfrastruktur verursachen. Dabei sorgt insbesondere der Datenverkehr zwischen den Domänen der einzelnen Netzbetreiber für hohe Kosten. Ansätze zur Reduzierung dieses Inter-Domain-Verkehrs sind deswegen ein aktueller Gegenstand der Forschung. Dabei erweist es sich als schwierig, gleichzeitig den Interessen der Nutzer und Netzbetreiber gerecht zu werden, da viele Lösungen zwar den Datenverkehr reduzieren, aber auch zu einer Verschlechterung der Dienstgüte für Nutzer führen können. Im Gegensatz dazu wird in dieser Arbeit ein neuartiger *anreizbasierter Mechanismus* entwickelt, um beide Seiten zufrieden zu stellen. Dabei bietet ein Netz-

betreiber den Nutzern Anreize in Form von *freier* zusätzlicher Bandbreite an, um ein kooperatives Verhalten zu belohnen und gleichzeitig die Leistungsfähigkeit des Systems zu erhöhen. Mittels Simulationen wird gezeigt wie ein intelligenter Einsatz von begrenzter zusätzlicher Bandbreite den Anforderungen der Inhaltsanbieter, Nutzer und Netzbetreiber gerecht werden kann. Der teure domänenübergreifende Datenverkehr kann erheblich reduziert und die Server der Inhaltsanbieter zusätzlich entlastet werden. Dieser Effekt tritt sogar ein, wenn der Mechanismus nur von einzelnen Netzbetreibern eingesetzt wird.

Zuletzt werden die Peer-unterstützten Inhaltsverteilsysteme aus der Sicht der Nutzer betrachtet, wobei neben der Internetverbindung (und der dadurch übertragenen Daten an andere Nutzer) insbesondere die Onlinezeiten von Bedeutung sind. Längere Onlinezeiten der Nutzer können für eine bessere Verfügbarkeit der Inhalte im System sorgen, wodurch die Server stärker entlastet werden. Allerdings führt dies auch zu niedriger Auslastung der Endgeräte der Nutzer und kann erhebliche (aber vermeidbare) Kosten für die Nutzer verursachen. Dieses Problem wird besonders deutlich beim Einsatz der sogenannten *Set-Top-Boxen*, wie zum Beispiel Digitalreceivern, Videorekordern und Spielkonsolen. Internetfähige Set-Top-Boxen werden bereits eingesetzt, um Videoinhalte von Servern zu beziehen, sollen aber zunehmend auch Inhalte bereitstellen können. Dabei wird oft von ständiger Verfügbarkeit solcher Geräte ausgegangen, was durch geringe Auslastung zu unnötig hohen Energiekosten führen kann. Um Energiekosten einzusparen ohne systemweite Leistungseinbrüche zu erleiden, werden in dieser Arbeit geeignete *Standby-Strategien* für Set-Top-Boxen vorgeschlagen. Die Leistungsfähigkeit dieser Strategien wird evaluiert und es wird gezeigt, dass hierdurch erhebliche Energieeinsparungen erzielt werden können. Es werden auch die Auswirkungen auf die Netzbetreiber berücksichtigt und eine Erweiterung entwickelt, um die Verfügbarkeit der Inhalte innerhalb einzelner Netzwerkdomänen zu gewährleisten. Dadurch werden die Anforderungen aller drei Parteien berücksichtigt.

Quality means doing it right when no one is looking.

— Henry Ford

ACKNOWLEDGMENTS

Writing a thesis is a complex and challenging task that requires a considerable effort from the author but also, at least as important, invaluable support from many other people.

First of all, I would like to thank Prof. Ralf Steinmetz for offering me an opportunity to write this thesis under his supervision. I would also like to thank my co-supervisor Prof. Phuoc Tran-Gia for his feedback, especially while finishing my work.

My special thanks go to my colleagues from the Peer-to-Peer Group with whom I have had many interesting and fruitful discussions and who have provided valuable feedback on this thesis. In particular, I am very grateful to Sebastian Kaune and Osama Abboud for the collaboration on research projects and publications.

Furthermore, my thanks go to my colleagues from the Multimedia Communications Lab at the Technische Universität Darmstadt and to the many partners from the SmoothIT project for the successful collaboration and support in various tasks.

I would also like to thank my students, especially Sebastian, Christian, Florian, and Markus. Your excellent work, valuable feedback, and critical questions helped me a lot in my projects.

Finally, I would like to express a large gratitude to my parents, Frieda and Alexander, my wife Ina, and my son Henry for their support and thoughtfulness.

Darmstadt 2011

CONTENTS

1 INTRODUCTION

Internet-based multimedia content delivery enables users to watch desired content from any location at an any point of time. With the increasing capacities of end-user devices and faster Internet connections, the popularity of such services is growing steadily. Some forecasts predict that by 2014 the share of video content will exceed 91% of the global consumer traffic, resulting in the equivalent of 12 billion DVDs per month to cross the Internet [35]. Figure 1 shows that *video streaming* will significantly outweigh other types of consumer Internet traffic, such as file sharing, Web, Voice over IP (VoIP), and online gaming. Contrary to file transfers, video streaming enables users to watch the video while downloading it, which imposes strict requirements on the delivery infrastructure. The users expect a performance similar to the traditional television with short startup delays and without performance degradations or playback stalling during watching. This is exacerbated by the growing requirements on video quality, such as higher resolutions and additional features (high-definition and 3D videos). The higher quality typically results in increased video bitrates that require higher download bandwidth.

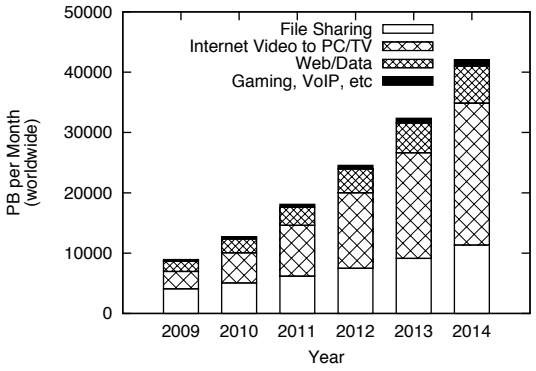


Figure 1: Forecast for global consumer Internet traffic (data from Cisco [35]).

Today users are increasingly able to consume videos directly from their TV screens using Internet-enabled Set-top Boxes (STBs) such as digital video recorders, game consoles, or other entertainment devices. Questions arise as which delivery architecture is able to provide this vast amount of video content to end-user devices and which mechanisms are required to make this architecture scalable and cost-efficient. Common solutions are centralized and decentralized delivery architectures employing various mechanisms to deliver video streams to end-users. These delivery architectures build *overlay* networks on top of the underlying Internet infrastructure.

The simplest architecture for video streaming is based on the centralized *client-server model*. Here (one or many) video servers send a separate video stream to each client, which results in high bandwidth costs for popular content and potential scalability issues for large numbers of concurrent users. Some studies state that YouTube, a very popular video-sharing website that uses a server-based delivery architecture, might have bandwidth costs of video delivery of around one million US dollar per month [65, 110].

The peer-to-peer (p2p) paradigm offers a promising alternative to pure server-based video distribution networks [139]. Here, the users, called *peers*[1], not only consume but

1 In the context of this thesis we are using the terms *peer* and *user* interchangeably.

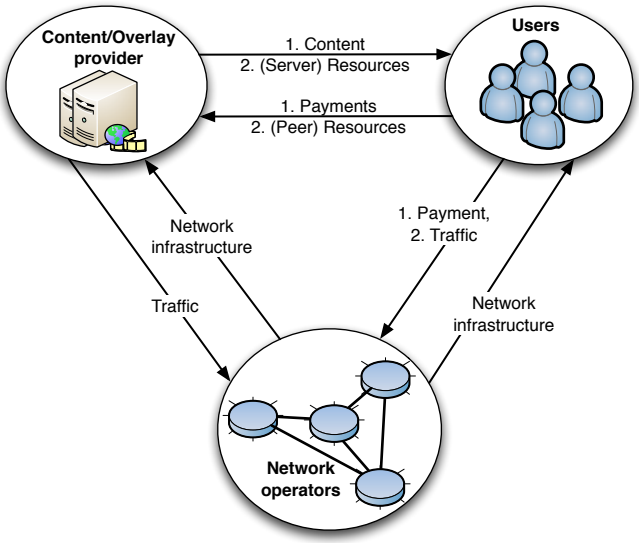


Figure 2: Relationships among the stakeholders of peer-assisted VoD.

also provide services to other peers. The application of the p2p paradigm to video streaming uses peers' resources, such as local storage, computational power, and bandwidth, to reduce the load and costs of content servers. In the extreme case of *pure p2p streaming*, there are no dedicated servers anymore and all services are provided by regular peers.

If we consider a *commercial* streaming system, a pure p2p solution turns out to be insufficient because it lacks important properties such as service guarantees for users, security, and control by the content provider [90]. In order to overcome these limitations, a *peer-assisted* architecture can combine content servers and peers intelligently.

In this thesis, we focus on peer-assisted Video-on-Demand (VoD) streaming, where users can request pre-stored videos at any time. Figure 2 shows different stakeholders participating in peer-assisted VoD. In order to understand the relationships between them and to identify the possible tensions, we must understand their roles in the architecture:

OVERLAY PROVIDERS contribute the initial content and host servers for content injection and indexing. In a pure commercial scenario the overlay provider also acts as a *content provider*[2], while in a scenario with user-generated content the content is contributed by users that upload it to content servers. Typically, an overlay provider receives certain payments for the hosted content, either directly from users (usage-dependent or subscription-based) or indirectly via advertisements.

USERS consume the streamed videos but also provide their resources to the system, such as upload bandwidth, local storage space, and online time. The users typically pay flat-rate fees for the Internet access, which explains why they allow an overlay provider to use their upload bandwidth.

NETWORK OPERATORS provide infrastructure for Internet access and receive flat-rate payments from the users for this service. Typical delivery overlays span several network domains controlled by different network operators. Therefore, network operators must manage both the internal and external traffic flows to avoid congestion and excessive payments for traffic transit.

2 From now on we are using the term *overlay provider* as a general term for the content and server provider.

A peer-assisted solution shifts the main load of content delivery from the overlay providers' servers to users. However, the actual delivery costs are shifted from the overlay providers to the network operators [65]. The reason is the widespread acceptance of flat-rate based pricing for the Internet access. These pricing schemes allow network operators to attract users and to sell high-speed Internet connections. But peer-assisted overlays can also lead to bottlenecks and link congestions, since the Internet architecture is built for the client-server traffic pattern where the traffic flows from content servers to the users. In the last years, the management of overlay traffic that crosses the boundaries of network operators' domains has gained a lot of attention in the research community [17, 114, 157, 33]. Thereby, various traffic management methods have been proposed to relieve the tension between the overlays and network operators. However, most of them fail to satisfy the demands of both parties.

In the following section, we take a closer look at the challenges of peer-assisted VoD addressed in this thesis.

1.1 CHALLENGES

In this section, we discuss the specific challenges of peer-assisted VoD streaming. Most of these challenges arise from the necessity to achieve quality of service (QoS) comparable to client-server systems, while using the limited resources of unreliable peers.

- *Limited resources* of an individual peer compared to a typical server mean that the resources of many peers must be combined to serve the same streaming request. This applies in particular to the upload bandwidth, which is typically much smaller than the download bandwidth [115, 39].

- *Heterogeneity of peers* in terms of their resources and behavior. For example, the upload capacity is a resource that differs between the users and affects the system's performance significantly [88]. The relevant behavior includes the request frequency and the time a peer stays online to serve other peers without consuming any content. The unpredictability of these factors makes a proper dimensioning of the servers (that must provide missing resources) difficult. While over-dimensioning might waste server resources and result in undesired costs, an under-dimensioned system will fail to provide the desired level of streaming quality.

- *Lack of service guarantees* at peers makes it difficult to ensure a quality streaming experience to the users (comparable to well-dimensioned server-based systems). In a commercial scenario, in contrast to pure p2p-based systems, all users should be able to receive the quality they paid for, which makes the coupling between the peer's contribution of resources and received streaming quality undesirable [119].

- *Missing or insufficient incentives for users to contribute their resources* are a common issue for p2p-based systems [81, 7, 96, 106]. In a commercial scenario, it can be partially solved by offering rewards or discounts for contributed resources [65, 106]. For example, a peer might get certain credits for each megabyte of data uploaded to other peers. However, this does not solve the issue of users that should remain online in order to provide content availability.

- *Energy consumption* is becoming an important challenge for content delivery [91]. While various approaches were proposed to increase the energy efficiency of servers and routers in terms of reduced power consumption [44], the same issue applies for the users' devices. One interesting aspect here is whether an idle peer should stay online to serve new requests or leave the system. While the first option would maximize the peer's contribution to the system, the second

option would save energy that might be wasted if the services of this peer are not required.

- *Negative impact on the network infrastructure* is another issue in peer-assisted and pure p2p delivery architectures. Most p2p and peer-assisted overlays apply their own application-level routing mechanisms that might have undesired effects on the underlying network such as congested links or high costs for the transit traffic [5, 133]. In this regard, existing approaches can advise the application of network-friendly decisions [134, 157, 124]. However, obeying them might deteriorate the overlay's performance [120, 93]. Nonetheless, some network operators try to enforce the fulfillment of their policies, which results in dissatisfied overlay users [145, 152]. This raises the need for alternative solutions able to satisfy the requirements of all parties: overlay provider, network operator, and users.

After presenting the challenges, we describe the goals of this thesis in the next section.

1.2 GOALS

The main question of this thesis is how to satisfy the partially conflicting interests of the three stakeholders involved in the peer-assisted VoD: overlay providers, users, and network operators. On the one hand, an overlay provider aims to reduce the costs of server hosting by utilizing user resources. However, too extensive reduction might deteriorate the streaming quality for the users. On the other hand, network operators wish the overlays to behave network-friendly. This could reduce the amount of costly transit traffic. However, network-friendly behavior might contradict the fulfillment of service guarantees to the users or even require the deployment of additional servers, which is undesirable from the overlay providers' perspective.

In this thesis, we aim to improve the peer-assisted VoD by considering the requirements of all three stakeholders, as follows:

1. The *first goal* of this thesis concerns the server hosting costs of overlay providers. In this regard, we aim to utilize servers and peers efficiently in order to minimize the load on the content servers while guaranteeing a desired QoS level.

2. The *second goal* is to reduce the large amount of costly transit traffic generated by peer-assisted (and pure p2p) overlays. The network operators and overlay providers should cooperate to reduce this traffic in a user-friendly way. This would enable network operators to attract VoD users while keeping their own costs low. At the same time, overlay providers and users could benefit from the higher overlay performance.

3. The *third goal* is an efficient management of such end-user devices as home gateways and STBs. Contrary to desktop PCs, these devices do not have to be manually controlled by the users and could boost the performance of peer-assisted VoD. The question remains how to utilize these devices efficiently from the users' point of view.

This thesis addresses the presented goals and provides adaptive mechanisms in order to achieve a suitable trade-off incorporating the interests of the involved stakeholders.

1.3 OUTLINE

The rest of this thesis is structured as follows:

Chapter 2 introduces the relevant background for this thesis. The chapter provides relevant knowledge about the alternative delivery architectures and introduces major

p2p-based streaming methods and topologies, including an example p2p VoD protocol. Furthermore, it elaborates on the topology and requirements of network operators, especially in regard to their co-existence with p2p-based delivery overlays. Finally, the chapter provides relevant background of STBs concerning their utilization in a peer-assisted VoD streaming.

Chapter 3 presents in detail the scenario of commercial peer-assisted VoD. First, our architecture is described, followed by the major assumptions of this thesis. Subsequently, the chapter addresses the problem statement to be answered in the rest of this thesis and discusses selected architecture components.

Chapter 4 presents our approach to server and peer allocation in peer-assisted VoD, including the related work on this problem, an analytical model, and our allocation mechanism. We present two specific allocation policies in detail, followed by their evaluation in various scenarios, including the application of proposed policies for STBs.

Chapter 5 introduces an incentive-based approach to traffic management of overlay applications. To this end, we discuss the related work on traffic management of p2p-based overlays, its shortcomings, and analyze the requirements for alternative approaches. Subsequently, we present our approach to incentive-based traffic management, called Highly Active Peer (HAP) promotion. The chapter discusses the required functionality: monitoring, selection, and promotion of peers. Finally, we describe our evaluation methodology and report the results obtained in the conducted evaluation study.

Chapter 6 covers the aspect of energy awareness for peer-assisted VoD based on STBs. This includes the motivation and shortcomings of existing approaches for STB utilization. The system model and the online time management problem, including common approaches, are introduced. Subsequently, the requirements for adaptive standby policies are stated and two alternative policies proposed. The devised policies are carefully evaluated and compared to alternative approaches.

Chapter 7 concludes this thesis by summarizing our findings and contributions. It further discusses potential directions for the future research in the context of this work.

2 BACKGROUND

This chapter describes the terms, definitions, and concepts required to understand the scenario and contributions of this thesis. Since our goal is to make video delivery over Internet more efficient, we start with a discussion of common delivery architectures. Subsequently, we focus on the basics of p2p streaming and present relevant methods, topologies, and an exemplary state-of-the-art streaming protocol. We proceed with an overview of Internet's infrastructure and its interplay with video delivery systems. Finally, we elaborate on specific properties of end-user devices used to receive and re-distribute streamed videos.

2.1 DELIVERY ARCHITECTURES

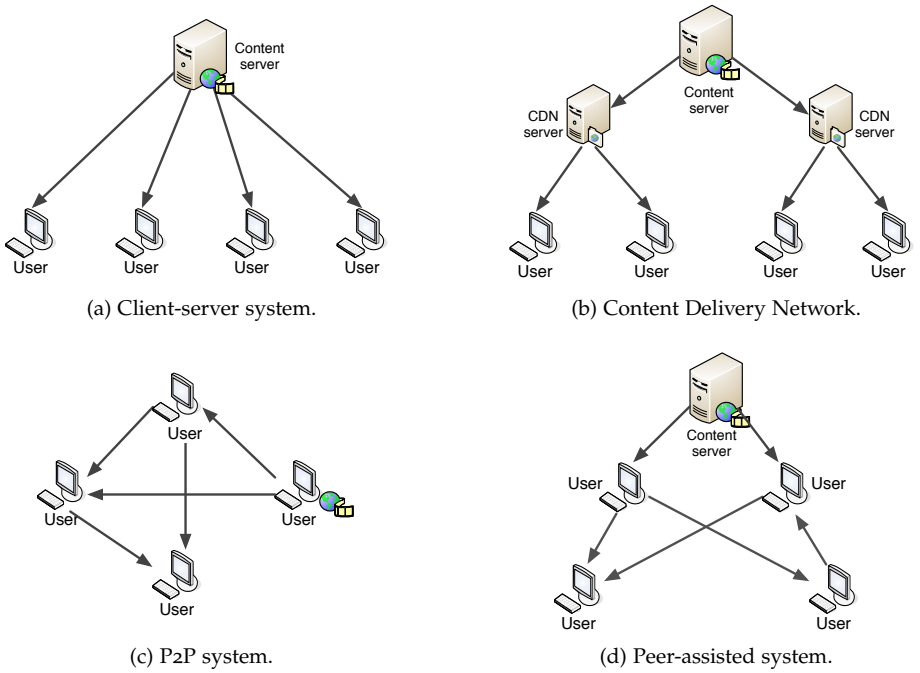
In this section, we discuss the basic architectures for video delivery over the Internet. The considered architectures build *overlay* networks on top of the underlying network infrastructure by autonomously deciding which host certain data should be delivered from. Thereby, we focus on application-layer delivery mechanisms that do not expose any additional requirements on the underlying networks, such as Internet Protocol (IP) multicast [49].

Initially, content delivery over the Internet was carried out in a *client-server* manner, where a content provider offered one or several central servers streaming content to clients (see Figure 3a). However, this approach exhibits limited scalability, since new servers must be added when the amount of users increases. Server-based systems are also costly because the servers must be dimensioned for the peak demand, which results in low server utilization and potentially high hosting costs [65]. An additional drawback is the potentially long distance from the client to the server resulting in long delay and extensive usage of network infrastructure [24].

To overcome the limitations of client-server architectures, Content Delivery Networks (CDNs) arose, which offer a distributed infrastructure of servers, with single servers (or server farms) located close to the users (see Figure 3b). A CDN replicates the content on different servers to maximize the bandwidth offered to the users and to avoid network bottlenecks close to the initial content servers [141]. Furthermore, a CDN typically hosts a vast amount of different content items resulting in a multiplexing effect by offering items with different popularity. By placing content close to the user, the latency and packet loss can be reduced, while the traffic between the network domains can be minimized. Some examples of commercial CDNs are Akamai[1] and Limelight Networks[2] that offer hosting services to content providers. Additionally, large content providers, such as YouTube[3], build their own CDNs [89]. Though a CDN improves content availability and avoids bottlenecks compared to pure client-server solutions, it still faces the issue of limited scalability and relatively high costs [67, 121].

Peer-to-peer (p2p) paradigm offers another method to deliver content by utilizing the resources of end-users (see Figure 3c). Here, each user, called *peer*, acts simultaneously as a consumer and as a service provider by uploading the downloaded content to other users [139]. This results in an increased system capacity with regard to the processing power, storage, and upload bandwidth [9, 99]. Unlike the client-server systems and CDNs, p2p systems scale well with the increased number of users [139]. However, pure p2p systems are not able to offer service guarantees common for

1 www.akamai.com (Accessed: 02.11.2010)
2 www.limelightnetworks.com (Accessed: 02.11.2010)
3 www.youtube.com (Accessed: 02.11.2010)



(a) Client-server system.

(b) Content Delivery Network.

(c) P2P system.

(d) Peer-assisted system.

Figure 3: Examples of delivery architectures.

server-based systems, since peers providing a service can fail or leave the system unexpectedly. This issue can be tolerated for file transfers by replacing the failing peers. However, in streaming systems this so-called peer *churn* might result in playback degradations [99].

Finally, *peer-assisted* content delivery systems illustrated in Figure 3d try to combine the benefits of server-based and p2p approaches [129]. Here, an overlay provider offers content servers to provide initial resources while peers contribute their local resources, mainly storage and upload bandwidth, so that the data can be streamed both from content servers and peers. On the one hand, such a system utilizes central servers to achieve availability, reliability, and quality of service. On the other hand, the system utilizes peer resources to reduce server hosting costs and to scale with the number of users [66]. Due to these properties, such systems attracted significant interest in the research community and several systems have been deployed [159, 158, 68, 97, 56].

In the rest of this thesis, we focus on the peer-assisted overlays. Since such overlays combine different types of entities, their management is more complex compared to traditional server-based systems. In order to understand how to combine server and peer resources intelligently, the next section presents basics and examples of p2p streaming.

2.2 PEER-TO-PEER STREAMING

Streaming is a common term for techniques that enable users to play content while downloading it. The basic principle is the same for audio and video content, while the latter case imposes much higher bitrates and, therefore, bandwidth requirements. Typical streaming over the Internet started with 200-400 kilobit per Second (kbps) encodings [66], while bit rates of one Megabit per Second (Mbps) and higher are considered as high resolution Internet videos [68]. In particular, the distribution of high resolution videos can result in high costs for content providers.

Due to the much lower bandwidth requirements of audio and to avoid possible synchronization problem of audio and video, we assume that both video and audio

are encoded into one *stream*. In order to provide the *watch-while-download* experience, streaming imposes at least the requirement of sequential (or near-sequential) download of file content and requires to use a video codec (and player) that enables playing partially downloaded video. One very popular video codec for video streaming on the Internet is H.264 [71].

In the context of this thesis, videos are streamed over the Internet using the Internet Protocol (IP), where sender and receiver processes are located on different hosts (e.g., streaming server and streaming client). A satisfactory streaming experience at the client side requires that the *startup* delay of video is relatively small (e.g., few seconds for short trailers and tenth of seconds for full-length movies) and the video playback does not stall.

2.2.1 *Methods*

In order to understand the challenges for peer-assisted (or in general for p2p-based) video streaming, it is important to understand how video content is generated and consumed. Two kinds of video streaming methods can be distinguished based on the data *freshness* [99]:

LIVE STREAMING offers one or more so-called channels. The video content is generated and/or streamed in real-time. Therefore, all consumers watch the same video scenes with only slightly shifted playback positions. Only the currently streamed channels and scenes can be watched and no rewind or seeking functionality is supported because future scenes are not available yet and past scenes are not stored on clients. A typical live streaming example is TV broadcasting, either over an IP, cable, or terrestrial network. The largest Internet-based (peer-assisted) live streaming network in Europe in 2009 was Zattoo [25].

ON-DEMAND STREAMING or Video-on-Demand (VoD) does not work with live data. Instead, a user has access to a certain amount of videos and can consume them at any time. During playback, the user can pause or jump forth and back to arbitrary positions. The latter might require additional playback delays if the data at the given position is not downloaded and buffered yet. One of the most popular VoD platforms on the Internet is YouTube that hosts (mostly short) user-generated content. There are also other VoD platforms focusing on full-length movies and TV programs, such as Hulu, BBC iPlayer, and Maxdome.[4]

In the following, we consider implications the different streaming methods have on a peer-assisted solution. For a better comparison, we also discuss file transfers as the simplest type of p2p-based content distribution. Indeed, besides the commonality of peers acting both as consumers and providers, there are important differences with regard to the download behavior for file transfers, live streaming, and VoD. Another commonality among the considered streaming methods (and file transfers) is the separation of a video into many *segments* that can be downloaded separately.

In a *live streaming* scenario, all peers have the same playback position and are therefore interested in the same content. This makes the usage of a multicast approach most promising. Furthermore, the time constraints are most sever in this scenario, since only very short distribution delay from the data source to the consumer is tolerable. This requirement limits the entropy of the system with regard to the choice of data blocks to be exchanged and selection of neighbor peers [48].

In case of *file transfers* the liveness of content is not important. This introduces significant freedom with regard to the order in which single segments are downloaded, such as the rarest-first policy that is used in the BitTorrent protocol [36]. The most

4 http://www.hulu.com/, http://www.bbc.co.uk/iplayer/radio, http://www.maxdome.de/ (Accessed: 02.10.2010)

important requirement here is a short download time of the file (or the average download speed, which is the reciprocal value of the download time). However, due to the near-random segment download order, for media files, such as video content, it is not possible to start playback before the whole file (or at least a major part) is downloaded. Therefore, file transfers are not considered as streaming of videos.

P2P-based *Video-on-Demand* offers a compromise between live streaming and file transfers. Contrary to live streaming, different users have different playback positions which makes multicasting less efficient [99]. On the other hand, if peers keep played video segments in their local storage, they can forward these segments to peers with earlier playback positions. VoD further allows some randomness with regard to the download order of video data. The data close to the playback position corresponds to the playout buffer and is typically downloaded in-order. If the data at the moving playback position is missing, the playback must be paused, which results in stalling (contrary to skipping of missing data for live streaming). The rest of the file, however, can be downloaded in an arbitrary order, which offers a lot of optimization opportunities [2].

A typical problem of streaming applications is the need to deal with unexpected delay or loss of required data to proceed with the playback. In a p2p system the peer churn (unexpected shutdown or failure of peers) as well as the limited bandwidth of peers often result in undesired delays for time-critical data. A technique called *prefetching* is used to build a reservoir of video to alleviate this issue [66]. To this end, the peer starts with the playback only after a receipt of a certain amount of consecutive video segments. Prefetching also allows to download the video with a higher speed than the actual video bitrate (the rate at which the data must be provided to the video player).

In this thesis, we are focusing on VoD streaming, though, many of the proposed techniques can be applied in a similar way to live streaming too. In the following, we take a look on typical topologies used for p2p-based (including peer-assisted) content distribution and compare their suitability for VoD streaming.

2.2.2 *Topologies*

The two basic schemes to support p2p streaming are *tree-based* and *mesh-based* approaches. Each of them organizes the structure of the overlay differently, which results in different properties with regard to the overlay efficiency, resiliency, and management overhead [4].

Tree-based P2P Streaming

In a tree-based approach, peers form a multicast tree with the initial content source (either a peer or a content server) as a root and regular peers as intermediate nodes and leaves. While, originally, this approach was mostly used for live streaming, certain proposals for tree-based VoD exist.

For example, in P2Cast [54] the peers with similar playback positions are grouped together. Each group is supported by a server with a separate base stream that is forwarded inside of the group in a multicast manner. Additional patch streams are used to supply peers with data missing in the base stream of their group.

An issue in tree-based streaming is the missing upload utilization of peers acting as leaves of the tree, a circumstance that some approaches try to alleviate by modifying the tree structure (e.g., multi-tree approaches [86, 23]). Another drawback of tree-based approaches is their vulnerability to churn since a failure of intermediate nodes disconnect whole subtrees and require repair of the tree structure [4].

Mesh-based P2P Streaming

Mesh-based systems utilize the swarming techniques well-known from the popular file-sharing protocol BitTorrent [36]. The video is again divided into segments, also called blocks or pieces by different authors. These segments with a typical size of 16–256 kilobytes are then exchanged between peers according to a certain download strategy [99].

Each peer regularly exchanges the list of completely downloaded segments with its neighbors so that each of them can request desired segments owned by this peer. If the segments owned by direct neighbors are sufficiently distinct a very high upload bandwidth utilization and download speed can be achieved [2]. However, this *diversity* requirement complicates the direct application of this approach to VoD. The reason is that segment diversity contradicts in-order download of segments close to the playback position of a peer [123, 2].

A typical drawback of a sequential download policy is low diversity of segment sets, since new peers have nothing to offer to peers with farther playback positions. On the other hand, a pseudo-random download order (e.g., the rarest-first policy used in BitTorrent-like systems) results in a very long startup delay. Indeed, the probability that a segment is missing in a playout buffer grows with the playout buffer size. In fact, it is $1 - (\frac{m}{n})^b$ for m downloaded segments out of n possible and buffer size b. Therefore, mesh-based p2p VoD typically applies a combined approach where segments closer to the playback deadline are downloaded with a higher probability [37, 149, 105, 2]. In the following, we present an example of a mesh-based streaming protocol.

2.2.3 *Give-to-Get Protocol*

The mechanisms, presented in this thesis, are applicable to a wide range of streaming protocols that share the features of prefetching and having a playout buffer. Nevertheless, a detailed evaluation must rely on a concrete system. For this purpose, we consider a state-of-the-art p2p streaming protocol. We used this protocol in one of our simulation studies (see Section 4.6) and in the prototype implementation of our resource allocation policies (see Section 4.7), which require insights into startup and stalling delays.

Give-to-Get (G2G) [105] is a mesh-based protocol for VoD streaming inspired by the BitTorrent protocol. G2G differs from the latter in several aspects: connection management, the bartering policy (the decision which other peers should be served by a given peer), and segment management. For example, the bartering policy in G2G lets uploaders prefer peers that turn out to be good forwarders. To this end, the uploader examines the amount of data each neighbor was able to forward to other peers.

In order to provide a watch-while-download experience, the video segments are managed in three distinct priority sets: high, medium (mid) and low (see Figure 4). The first one is the most crucial, since it corresponds to the playout buffer. Therefore, a peer always tries to fill the high priority set first. If it is already full, the mid priority set is filled, and, finally, the low priority set. Furthermore, the segments in the high priority set are downloaded in-order to avoid playback stalling. The other sets are filled according to the rarest-first policy to increase segment diversity in the neighborhood. Since the mid and low priority sets do not overlap with the playout buffer of the video player, these sets are filled to enable prefetching of video content. While the prefetching increases peer utilization, it might also result in bandwidth wastage. For example, if the user decides to stop watching content and leaves the overlay, the prefetched segments would be neither consumed nor forwarded to other peers.

Unlike the original G2G approach (as described in [105]), we do not skip the delayed segments of the video, but rather stall the playback until the playback buffer can

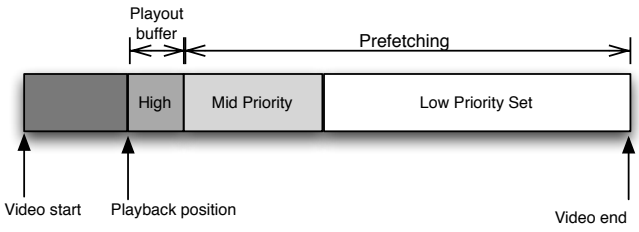


Figure 4: Priority sets of the G2G protocol (based on [105]).

be filled. This appears to be more suitable for a VoD application, since most users would prefer to see parts of the movie with delay instead of skipping them completely. Furthermore, stalling enables to offer these segments to other peers later on. The same solution is applied for VoD in Tribler [147], a prototypical implementation of G2G.

Additionally to the open-source Tribler client [147], G2G is used by the P2P-Next project that is building a p2p-based Internet Protocol TeleVision (IPTV) technology [75]. Since G2G was designed as a pure p2p protocol, it offers only limited support for server utilization. Tribler (and also the P2P-Next prototype) focus on pure p2p streaming in the first place. Typically, a powerful peer that possess a complete video copy, the so-called *initial seed*, is used to inject content in the first place. However, the upload resources of the initial seed do not adapt to the user demand. Instead, a fixed upload bandwidth is used to serve other peers. In a commercial scenario, when a content provider is offering such an initial seed, its non-adaptive nature might result either in unnecessary high upload costs for the content provider or bad streaming experience for the users. In Chapter 4 we develop an appropriate solution to this issue.

2.3 NETWORK OPERATORS

Content distribution of video content, as discussed in the previous section, became possible in recent years because of the steadily increasing bandwidth available to the end-users [6]. With today's download bandwidth for DSL and cable connections being at least one megabit per second, high quality videos can be streamed over the Internet. An additional factor is the widespread acceptance of flat-rate payment schemes without traffic volume limitations, which became de-facto standard in many countries [113].

All this factors enabled the emergence and rise of various content distribution methods discussed in Section 2.1 (server-based, CDNs and p2p-based content delivery). Since each method creates a *virtual* network on top of the actual network structure, the term *overlay* is used to distinguish it from the underlying (IP-based) infrastructure, which is consequently termed *underlay* [1]. One of the properties of many overlays is the lack of knowledge about the structure and link capacities of the underlay, which makes the overlay operation often suboptimal from the underlay's perspective [79]. Furthermore, especially the p2p-based content delivery tends to shift the costs from content providers to the network operators (not users because of flat-rate pricing) [65, 94, 120].

2.3.1 *Domains and Interconnections*

In order to understand the impact of video delivery overlays (which constitute major part of total Internet traffic [35]) on the underlying network, we must consider the structure of the Internet. The video consumers connect to the Internet via dial-up, cable, DSL, wireless, or dedicated lines. These connections are offered by *network*

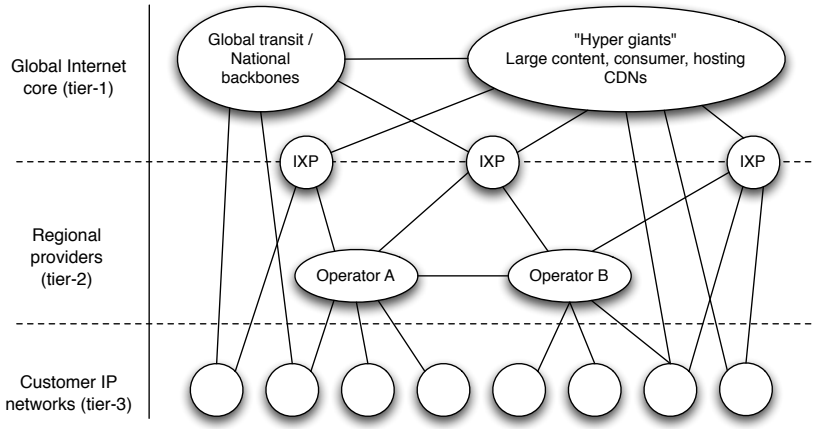


Figure 5: Emerging logical topology of the Internet (based on [89]).

operators that are also called Internet Service Providers (ISPs). The Internet as such constitutes thousands of edge networks, so-called Autonomous Systems (ASs), which are interconnected directly or via additional networks and backbones [89, 62]. A certain operator might own one or several ASs that might be located close together or be separated geographically, effectively constituting an operator's *network domain*. In the rest of this thesis, we address such domains as an organizational and business unit, especially when considering their inbound and outbound traffic that result in associated interconnection costs.

Network operators can be roughly classified into three hierarchical categories, so-called tiers, as shown in Figure 5, with the following characteristics [89, 61]:

TIER-1 network operators build the core of the Internet. They provide transit services over national and international backbones, with mostly tier-2 network operators as customers. While pure tier-1 operators do not provide services directly to end-users, large CDNs, such as YouTube and Akamai, build their own global networks that connect to the network operator domains at so-called Internet eXchange Points (IXPs).

TIER-2 network operators are connected to tier-1 network operators and rely on their transit services. Tier-2 network operators can manage their own (national or regional) backbones and provide services to tier-3 network operators and end-users.

TIER-3 network operators are typically smaller than tier-2 network operators (with regard to the network size and customer number) and rely on transit services of tier-1 and tier-2 operators to offer Internet connectivity to their customers.

With regard to the economics of network interconnections, certain payments are involved when traffic flows between operators' domains [65]: Typically, network operators closer to the Internet core (i.e. tier-1 and tier-2 operators) charge the network operators in the higher tiers (tier-2 and tier-3) for the exchanged traffic. This results in inter-domain traffic being *more* expensive for most network operators, a sever issue with overlay applications spreading over several operator domains and generating a lot of transit traffic. Because most of the end-users tend to generate more download than upload traffic (see e.g. [31]), *inbound* traffic from lower tiers dominates the interconnection costs of tier-2 and tier-3 network operators. The so-called 95-percentile rule is applied to charge for this traffic [79, 112, 113].

While the actual costs of intra-domain and inter-domain traffic depend on the type of the last mile connection (DSL, cable, or wireless) and the individual interconnec-

tion agreements between the neighboring domains, in the following, we assume the following simplified model (cf. [89]):

- Tier-3 network operators try to reduce their *inbound* traffic in the first place rather than the outbound traffic. This asymmetry stems from symmetric inter-domain links and asymmetric end-user access profiles.
- Tier-2 network operators try to reduce *inbound and outbound* traffic to/from tier-1 network operators. At the same time, they try to increase or preserve outbound traffic to tier-3 network operators.
- Tier-1 network operators try to maintain or even increase the inbound and outbound traffic, though try to avoid bottlenecks during busy hours.

2.3.2 *Coexistence with Overlay Networks*

From the arguments, presented in Section 2.3.1, we make two important observations: (1) The increasing popularity of high-resolution video content enables network operators to attract new customers and to sell high-end Internet connections [31]. (2) A content delivery overlay might span users and servers from various network operator domain and, therefore, induce significant amount of costly inter-domain traffic [89].

Consequently, the network operators are often interested in reducing the negative impact of overlays by various mechanisms. The relevance of this issue manifests in the recently created IETF working group for Application-Layer Traffic Optimization (ALTO) [134]. In this regard, an important concept is the *locality promotion*, meaning that peers should exchange data with peers inside of the same network domain instead of remote peers. This and other approaches aim to reduce the amount of inter-domain traffic. We discuss them in more detail in Chapter 5.

In this thesis, especially the requirements of tier-2 and tier-3 network operators are considered, since they are the only ones to interact with P2P users directly. Here, traffic management techniques can be applied by the overlay applications to act network-friendly (e.g., within a tier-3 network operator by reducing inbound traffic).

2.4 SET-TOP BOXES

Finally, we address the new kind of end-user devices that can be utilized for peer-assisted video streaming.

In the past, most Internet multimedia users downloaded and watched streamed content using their Personal Computers (PCs). With the increasing amount of IP-enabled entertainment devices, such as TVs, Digital Video Recorders (DVRs), and digital receivers, more content is being consumed without a PC. For example, the Cisco Visual Networking Index forecasts that the amount of traffic induced by *Internet Video to TV* will grow by 107 % per year until 2014 [35]. This content will be delivered via the Internet but viewed on a television screen using so-called Set-top Boxes (STBs).

Since STB is a broad term, different devices belong into this category. Conventional STBs are receivers for dedicated channels, such as cable networks, satellite, or terrestrial broadcasts. Additionally, IP-enabled STBs receive video data over the Internet or the local area network. STBs can also employ an internal hard drive and act as a Digital Video Recorder, where a user can download content once and then watch it at will. Simple functionality can be even provided by *home gateways*, such as DSL modems with a hard drive and extended features. More powerful devices, such as media servers or video game consoles, can offer more complex functionality and higher performance.

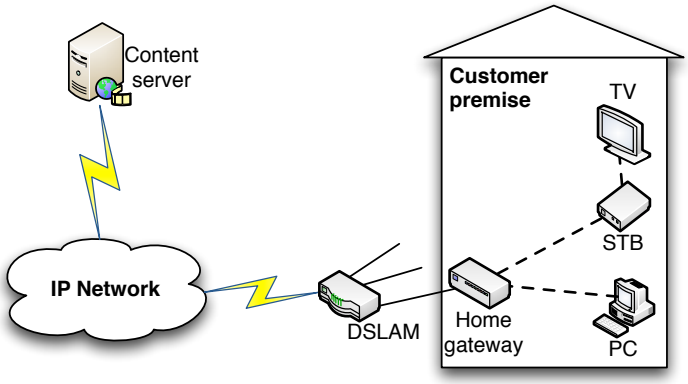


Figure 6: Typical multicast IPTV service architecture (based on [24]).

Different types of STBs are already used for various IPTV and VoD platforms such as Tivo, Apple TV, Roku player, and Sony Playstation.[5] In this regard, the research community and industry propose to use them also for peer-assisted video delivery [90, 24, 148, 41, 140, 85]. For example, Laoutaris et al. argue that STB-based architectures, which they call nano data centers, can benefit from higher availability and provider-side control. This should allow them to outperform traditional best-effort p2p systems where users' contribution is unpredictable and unstable [90].

2.4.1 *Deployment Alternatives*

A typical commercial IPTV architecture is shown in Figure 6. In this example, the video content is delivered from TV stations to end-users' devices (either PC or television via STB) through the IP backbone of one or several network operators. Digital Subscriber Line Access Multiplexers (DSLAMs) aggregate traffic for hundreds or thousands of users and, therefore, constitute a potential bottleneck with regard to the throughput. In a traditional server-based architecture, live content can be delivered via IP-multicast while VoD content must be delivered via single streams. In both cases, peer-assistance can be used to reduce server load and even to reduce the amount of inter-domain traffic [90, 24, 27].

In general, two major types of suitable end-devices can be considered in this architecture:

HOME GATEWAYS are networking devices such as DSL or cable modems used to connect home devices to the Internet. High-end gateways can be equipped with hard or flash drives able to store several hours of video data. In a peer-assisted architecture, such devices must be able to forward stored video content not only to the local STB, IP-enabled TV, or desktop PC but also to other users [148]. One of the most prominent properties of such devices is their high availability since they must be online in order to provide Internet connectivity to other home devices.

SET-TOP BOXES in the narrower sense are always connected to a television to display content. Typical examples are IP-enabled digital video recorders, digital receivers, and also increasingly game consoles. These devices exhibit stronger capabilities with regard to storage (they can have internal hard drives), CPU power, and main memory. These properties, as well as their convenient usage to

5 http://www.tivo.com/, http://www.apple.com/appletv/, http://videostore.de.playstation.com/, http://www.roku.com/ (Accessed: 12.10.2010)

display video content, make them attractive candidates for peer-assisted content distribution [74, 142, 27]. A potential downside is their relatively high power consumption (especially of game consoles) and, accordingly, users wish to turn them off in order to save baseline power [63, 91, 21]. This issue reduces the potential online time of such devices to few hours per day, therefore, reducing their availability and potential to offload content servers in a peer-assisted deployment.

2.4.2 *Open Issues*

Depending on the abilities of a certain STB, employing it in a peer-assisted solution results in different advantages or limitations. For example, a streaming-enabled DSL modem has limited CPU and storage capacity but also lower power consumption compared to a dedicated multimedia STB or game console. In general, most of such devices are able to support customized p2p-based streaming protocols as demonstrated by the P2P-Next project offering an STB-ready prototype of their software [140].

With the increasing complexity of STBs also their baseline power consumption increases. Even power-efficient devices that consume only 10-20 Watt in an idle mode, might generate significant annual energy consumption. Therefore, users might switch them off to avoid baseline power consumption [148, 91]. Such a selfish behavior, however, would reduce the efficiency of a peer-assisted system by making local content replicas unavailable to the system. Therefore, the overlay provider must consider the online time of STBs and the incurred power consumption as additional user costs.

The trade-off between the availability of STBs and their aggregated energy consumption is one of the research questions, covered in this thesis (see Chapter 6). We aim to find a proper compromise between the requirements of the overlay provider (low load on content servers) and users (high quality-of-service, no wastage of energy). Also the aforementioned requirements of the involved network operators are addressed.

2.5 SUMMARY

This chapter provided the technological background of a peer-assisted VoD system. We covered common architectures for video delivery on the Internet, namely: client-server systems, CDNs, pure p2p systems, and peer-assisted systems. We further discussed different methods to provide video content to users: VoD, live streaming, and file transfers. This background is relevant to understand the specific requirements of peer-assisted VoD that is the focus of this thesis. This chapter also justified the usage of a mesh-based topology and utilization of servers as initial and backup content sources. We further highlighted the tension between network operators and delivery overlays, where operators would like to limit their interconnection costs, while still allowing high-quality video delivery to attract customers. Finally, we introduced STBs as promising devices to build peer-assisted streaming platforms and stressed the power consumption of idle devices as a relevant cost factor.

The next chapter describes the considered architecture for peer-assisted VoD, our main assumptions, the problem statement of this thesis, and selected components of the considered architecture.

3 SCENARIO

In this chapter, we present an architecture for peer-assisted VoD. The architecture combines centralized and decentralized components in order to operate cost-efficiently but with service guarantees. The considered system utilizes state-of-the-art components such as the underlying streaming protocol, network-friendly resource selection, and IP-enabled STBs. In this sense, the architecture involves all three stakeholders of peer-assisted video delivery: the overlay provider, network operators, and users. In the described scenario, the mechanisms devised in this thesis are applied. Furthermore, this chapter presents the assumptions and problem statement to be solved by the mechanisms presented in the rest of this thesis. We also discuss selected components of our system.

3.1 ARCHITECTURE

The considered architecture is inspired by the rising popularity of commercial peer-assisted VoD systems such as PPLive, SopCast, and the BBC iPlayer[1]. These systems combine central servers used to guarantee basic content availability with *peer-assistance* (in terms of upload bandwidth and storage) to reduce costs of video delivery and to provide scalability for a large number of users. However, the proposed approach differs from existing systems with regard to the utilized protocol for server allocation and focuses on the collaboration with network operators and STB support. In the past, discontinued peer-assisted applications such as Joost have shown that building a successful application is not trivial [127].

The considered peer-assisted architecture comprises three types of entities (see Figure 7): indexing server, content server(s), and peers. These entities constitute the delivery overlay and are spread over multiple network domains managed by different operators. Overlay provider maintains indexing and content servers, but also manages the client application (including the software development, configuration etc.) and, therefore, controls the mechanisms applied by the application.

Video files are divided into segments and initially injected from content servers. These servers upload segments to a subset of peers that exchange these segments with each other and, therefore, contribute their upload bandwidth.

The *content servers* initially inject new content into the system and act as backup video sources in case the desired content cannot be streamed from peers. This backup functionality is unavoidable to deal with unpredictable user demand, however, the load on these servers must be minimized to reduce hosting costs.

The *indexing server* tracks videos available in the overlay, including the information about videos downloaded, and provides information about available video sources (both servers and peers) to peers requesting them. To this end, the indexing server answers a search request with a list of peers holding video replicas, which enables bootstrapping and effective contact management for participating peers. While the indexing server has the global view of the system, the information might be partially outdated.

Finally, the *peers* download videos on behalf of their users', store videos in (size-limited) local caches, and offer videos to other peers. The peers are organized in a mesh-based topology that is resilient to failures of single peers and can easily support upload of the same video from multiple sources. Organizing peers in a mesh instead

1 www.pplive.com, http://www.sopcast.org/, http://www.bbc.co.uk/iplayer/radio (Accessed: 03.11.2010)

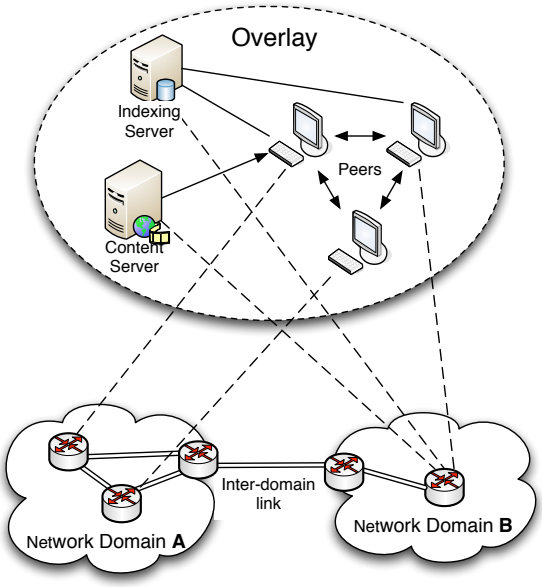


Figure 7: Simplified example of peer-assisted overlay, which shows the involved entities: servers, peers, and network routers. Dashed lines show which router certain peers and servers are attached to.

of a tree, simplifies the handling of node arrivals and departures (as discussed in Section 2.2.2).

A new peer joining the system selects a video to watch and sends a request to the indexing server. The returned peer list contains addresses of peers owning at least some segments of the video. The peer connects to some of other peers that build its *neighborhood*. Peers exchange the information about available and desired segments with their neighbors and choose segments to download depending on their playback deadlines, that means, segments closer to the current playback position are prioritized (see Section 2.2.3). The most urgent segments are managed in a high priority set, whose size can range from few seconds to a minute of the video. Since the high priority set roughly corresponds to the playout buffer of the video player, peers are interested in keeping it full.

3.2 ASSUMPTIONS

In this section, we present our main assumptions with regard to the behavior and goals of stakeholders participating in the presented architecture.

3.2.1 *Costly Server Traffic*

Our first assumption is that the overlay provider aims to reduce the load on its servers. In order to satisfy user demand, the overlay provider must dimension its server for the peak demand (typically in the prime evening hours), a fact that makes server hosting expensive [65]. We consider the common case when the overlay provider pays for the uploaded traffic volume [66]. While peer-assistance enables reducing this volume, the savings depend on the efficiency of the utilization of peer resources.

A peer-assisted system is highly dynamic because peers might arrive and depart from the system unexpectedly, which results in fluctuating demand *and* supply of bandwidth. Additionally, the resources of users are heterogeneous: some users have

high bandwidth capacities and others quite low [39]. A departure of a high bandwidth peer can result in a significant performance decrease [88]. Even worse, since the upload bandwidth is typically much lower than download bandwidth, only a subset of peers exhibit upload bandwidth higher than the video bitrate [39]. Thus, the dynamic number of concurrent viewers and fluctuating capacities of peers must be alleviated by servers' contribution. This results in a potentially high server traffic and, therefore, high hosting costs.

3.2.2 *Streaming Quality*

We assume that users of a commercial peer-assisted VoD system expect streaming quality similar to that of a well-dimensioned server-based system [10]. The question arises how to measure this quality. There exist various metrics to assess the user's Quality of Experience (QoE), including subjective and objective metrics [165, 45]. Unfortunately, subjective metrics require real users to provide perception feedback online and are hardly applicable to simulation studies. On the other hand, objective metrics work at the frame level, which limits the scalability of evaluation studies. To avoid such constraints many researchers resort to more network-centric Quality of Service (QoS) metrics, such as the continuity index [149, 97], throughput, or packet delay [60].

For our purpose, VoD-related metrics: *startup delay* and *total stalling time* appear as most appropriate. They are objective and can be directly perceived by users. Ideally, the video playback should start within few seconds and there should be no playback stalling. The exact values of tolerable startup delays and stalling times depend on the content type and preferences of single users. For example, a startup delay below ten seconds is tolerable and the relative stalling should be less than one percent of video length.

3.2.3 *Incentives for User Contribution*

We assume that the overlay provider is able to incite or enforce users to contribute their resources to the system. The more a peer contributes the higher reward should be granted by the overlay provider. This can be most easily measured in terms of useful data uploaded to other users.

The key to a high contribution is the provision of peer resources, especially the local storage (to cache videos), online time (to be available), and upload bandwidth. With regard to the local storage we assume that each peer is willing to cache at least few videos at once, which requires a moderate storage space of 10-20 gigabytes (GBs). For the online time, either the level of altruism or strategic behavior can dominate the decision whether a peer is online or not. In any case, being online without contributing any resources is highly undesirable, since such an idle peer wastes resources without any reward from the overlay provider.

Finally, we do not consider a reputation scheme or special incentive mechanism to make users contribute their resources. There is extensive research on this topic in the p2p community (e.g. [47, 82, 119, 81]), but for the considered commercial scenario we assume that users are indeed interested in contributing more (as far as their resources suffice and the overhead is not too high, e.g., staying online 24h to serve just one GB of data would be inappropriate).

3.2.4 *Bandwidth Bottleneck*

The next assumption is that peers' *upload bandwidth* is a crucial bottleneck in peer-assisted VoD. In the best case, the upload bandwidth of an average Internet connection

is similar to the desired bitrate of a video in moderate quality. For example, according to the Ookla database [115], in 2010 the mean upload bandwidth of a residential Internet access in Germany was 1.4 Mbps (vs. 15.42 Mbps for the download bandwidth). At the same time, according to Cisco [35], a high definition Internet video might exhibit a bitrate of up to 31 megabyte per minute (= 4.13 Mbps). Consequently, in order to provide videos in high definition quality, several peers must contribute their bandwidth or even the content server must help them.

This assumption is consistent with the trends observed in commercial VoD systems where it might require 6–8 peers to serve one video stream [68]. The reason is that the users orient their expectations on their *download bandwidth* that is typically much higher than the upload bandwidth.

3.2.5 *Costly Inter-Domain Traffic*

Next, we consider the costs of network operators that carry the traffic of video delivery. Despite their interest in customers using such delivery overlays, the operators must operate their networks cost-efficiently. Unfortunately, the flat-rate payment plans for operator's customers do not allow for usage-based charges and, therefore, non-monetary techniques are required.

We assume that the interconnection costs dominate overlay-induced costs of network operators (see also Section 2.3). Consistently, the operators aim to reduce the traffic volume that crosses the boundaries of their domains (the same assumption is made for example in [79, 150, 112]). Furthermore, network operators are continuously upgrading their networks with new technologies, such as fiber-to-the-home [31] to catch up with the increased user demand for higher bandwidth. This might lead to an over-proportional increase in the access bandwidth and create bottlenecks in the core of the Internet.

3.3 PROBLEM STATEMENT

In this section, we state the questions arising from the assumptions discussed in the previous section. These questions constitute the problem statement of this thesis:

REAL STREAMING EXPERIENCE with startup delays of few seconds and almost zero stalling with high probability (for example, to guarantee certain delay for 95% of users). Since we cannot guarantee that peers can always provide this quality level, servers must be intelligently allocated to assure quality.

REDUCED LOAD ON CONTENT SERVERS compared both to the pure server-based and state-of-the-art peer-assisted solutions. This should happen by intelligent utilization of *available* peer resources instead of server resources. Note that the overlay provider does not have global knowledge of peers' capabilities and states. Therefore, the estimation of required server resources is more complex than in a client-server scenario.

REDUCED INTER-CONNECTION COSTS for network operators should be achieved through overlay- and network-friendly traffic management techniques. In this regard, *strategic* behavior of overlay providers, users, and network operators is assumed, which means that each stakeholder tries to optimize the own payoff [120, 11, 113]. This means that any cooperative solution that does not benefit both sides, the overlay and the network, will be hardly accepted.

REDUCED ENERGY CONSUMPTION OF PEERS should be achieved in the STB-scenario (which can be optionally applied in the system). This addresses the elimination of idle times when peers stay online without contributing any resources. But it

also demands to avoid that peers with desired content go offline, which requires content servers to provide content that could be otherwise provided by peers.

The ability to fulfill the above requirements results in a content delivery solution capable to satisfy the demands of all three stakeholders (overlay provider, users, and network operators), which is the so-called *triple-win* situation for Internet-wide content delivery [138].

3.4 SELECTED COMPONENTS

In this section, we describe several components of the considered architecture to complete our scenario. The understanding of their functionality is relevant for the analysis and evaluation of the mechanisms devised in this thesis.

3.4.1 *Streaming Protocol*

The efficiency of the utilized streaming protocol affects both the streaming quality and the load on content servers. Because of the large amount of proposed streaming protocols, we do not design a new one from the scratch but rather apply our mechanisms to a state-of-the-art protocol. The level of maturity, documentation, and availability of a prototype where we can integrate and test our mechanisms dictate our choice. In the default case, we use the Give-to-Get streaming protocol described in Section 2.2.3. An open-source implementation of this protocol enables performance analysis of our solutions in a realistic setting both via simulations and prototype measurements (as done in Chapter 4). Whenever appropriate, we abstract from the details of the underlying streaming protocol, thus, focusing on its generic properties.

Furthermore, we use the same protocol for data exchange from server to peers and between single peers. This simplifies the integration of different entities, including regular desktops, content servers, or even STBs.

3.4.2 *Server Utilization*

There are several methods how server resources can be utilized in a peer-assisted system:

1. *Static resource allocation per peer* means that servers provide guaranteed bandwidth to each peer, with other peers offering only patch streams for future content (as proposed e.g., by Guo et al. [54]). In this case, peers would only upload data outside of the high priority set. A problem arises since this approach may under-utilize peer resources and waste server resources.

2. *Static global allocation* lets servers upload only most urgent segments *if they cannot be served by peers* as proposed e.g. in [2]. However, this simple approach might request urgent segments from servers even though they could be served by other peers. Moreover, it would be difficult to decide how many peers each server should serve due to fluctuating necessity of server contribution.

3. *Static allocation of resources per swarm* assigns fixed server resources to a single video (where all peers watching this video constitute a single *swarm*). Because of the fluctuating demand of video popularity (e.g. flash crowd effects), a precise estimation of user demand is difficult and can be applied only for short time spans of stable demand. Furthermore, this approach requires prediction of content popularity. We consider this approach as our reference scenario.

4. *Dynamic allocation of resources* is the most flexible option that enables adjusting the server contribution depending on the current situation in the system. Be-

cause of its flexibility and the opportunity to guarantee stable performance, we follow this approach and design suitable algorithms (presented in Chapter 4).

3.4.3 *Content and Peer Search*

In order to exchange video segments a peer must first discover interesting content and then find peers or servers having this content in their caches. This content and peer search can be implemented either using a central entity (such as a tracker in BitTorrent) or with a decentralized search mechanism.

In a commercial scenario, the resources required to provide a search functionality are much lower than those required for the actual delivery of video content (we show an example in Section 4.6.2). Furthermore, for the purpose of accounting and statistics collection, an overlay provider must know which users consume which content. Therefore, our architecture uses a centralized search mechanism in the first place with a centralized *indexing server*.

Optionally, a decentralized search mechanism (such as gossiping [30] or distributed hash tables [43]) can be used to offload the indexing server, but in general the search accounts for much lower traffic than the delivery of video streams, and is, therefore, not the main cost factor for the overlay provider.

3.4.4 *Peer Selection*

In its general sense, *peer selection* refers to the selection of a subset of peers out of a candidate set according to a certain criteria, such as current load, network proximity etc. A typical example of peer selection is the choice of the overlay neighborhood that is the set of peers a given peer connects to. Another example when peer selection becomes necessary is when the available upload bandwidth of peers caching a certain video becomes larger than the required service rate. By utilizing this degree of freedom different criteria can be considered, such as load distribution, network-friendliness, or reputation of peers [123].

In peer-assisted VoD, the same video segment can be downloaded from different peers. The selection of a suitable peer to provide these segments can influence the performance of the overlay and the load on the network infrastructure. In the following, we consider examples of simple peer selection metrics. Then we show how they can be combined.

- *Random peer selection* is the simplest metric where peers are chosen randomly. Underlay-aware metrics sort peers according to the number of inter-domain hops (the amount of network domains that must be crossed to send data from the source peer to the current peer) or end-to-end delay [1]. Typical proximity metrics are very crucial for live streaming to assure data freshness, while in a VoD scenario the average throughput is more significant than the end-to-end delay of individual data packets.

- *Load balancing* is a suitable metric to increase fairness among users (with regard to their contribution to the system). It can also improve utilization of peers that cached less popular content. Given two candidate peers p_1, p_2 and their contribution to the system denoted as $u(\cdot)$, we define:

$$p_1 <_{load} p_2 \text{ iff } u(p_1) < u(p_2). \tag{3.1}$$

  A peer can now sort the candidate list in *ascending* order according to this relation and send download requests to top peers. This will result in peers with less contribution so far to serve new download requests.

- *Locality awareness* [164] is an important metric because of the interplay with the underlying network (see also Chapter 5). Here, the peers from the *same network domain* as the requesting peer p_0 should be preferred. With $d(p)$ being the network domain of peer p, we define:

$$p_1 <_{loc,p_0} p_2 \text{ iff } d(p_1) = d(p_0) \text{ and } d(p_2) \neq d(p_0). \tag{3.2}$$

  That is, by sorting the peers in ascending order the peers from the same domain will be selected first.

While Equation 3.1 aims to reduce the number of bandwidth misses by offloading peers that cache mostly popular content, Equation 3.2 aims to reduce the amount of inter-domain traffic. Since many candidate peers might be located in the same network domain, a combination of locality awareness with other metrics appear feasible. In this regard, we propose a combined metric where locality awareness works as a filter for another metric (such as the load balancing metric) by following the rule: Given a set of potential candidates order the list according to network domain as a primary key and totally served data as a secondary key. The resulting formula is:

$$p_1 <_{combined,p_0} p_2 \text{ iff } (p_1 <_{loc,p_0} p_2) \text{ or } (p_1 =_{loc,p_0} p_2 \text{ and } p_1 <_{load} p_2). \tag{3.3}$$

Note that $p_1 =_{loc,p_0} p_2$ always holds if $d(p_1) = d(p_2)$ (independent of p_0). The impact of these peer selection metrics on the system performance is covered in Chapter 5 and Section 6.5.4.

3.4.5 *Caching Strategy*

In our scenario, each peer can store several videos in its local cache. A *cache replacement policy* is involved if the maximum cache size is exceeded. This policy aims to reduce the number of system-wide cache misses by adjusting the amount of available copies to the request rates of individual videos. Commonly used *cache replacement* policies, such as Least Frequently Used (LFU) and Least Recently Used (LRU) are good candidates to be used. They can rely on either local request history (managed by each peer) or global history (managed centrally at the indexing server). Neighborhood-based approaches are also possible, where peers exchange historical data with their neighbors.

As shown by the related work [144, 156], even simple cache replacement strategies, such as LFU, LRU, or even First-In First-Out (FIFO) work quite well for peer caches. Since each peer acts as a cache, the system-wide number of cached copies per item is proportional to the popularity of the content [144]. Nevertheless, in certain systems, more complicated replacement schemes are used [68]. We experimented with various schemes (combining LFU, LRU, or FIFO with local or global request history) and chose a simple policy (FIFO) that relies only on *local* information at a peer and avoids additional messaging overhead, because more complex policies did not result in significantly better cache hit rate.

Replication is another alternative to assure appropriate availability of content in the system [148]. The difference to caching is that replication generates copies proactively. For example, with STBs content servers could distribute new content to STBs during idle hours so that they can be served to requesting users during busy hours. The major benefit of replication is traffic *smoothing*, since the server can shift the peak requests from the busy to idle periods. Nonetheless, the *total amount of data* that must be served by servers is not reduced by this approach. Even worse, if the prediction of the content popularity is wrong, the number of video replicas might be too high leading to an unnecessary usage of server resources. Because of this, we do not consider active replication in our system, though it can be easily integrated. Another possibility is that peers replicate the content of their local caches under certain conditions (e.g. each time they have to remove an item), but this again might not result in significant performance improvements [144].

3.5 SUMMARY

This chapter presented the overall architecture of the considered peer-assisted VoD system. The hybrid architecture combines indexing server and content servers as centralized entities with decentralized peers. We further described our assumptions with regard to the requirements of involved stakeholders such as high costs of inter-domain traffic and streaming-related quality metrics. The problem statement of this thesis was derived from the presented requirements and addresses all three stakeholders: overlay providers, network operators, and users. We further presented selected components of our architecture: the utilized streaming protocol, server utilization method, search mechanism, peer selection, and caching strategy. We use them as a foundation and evaluation setup for the mechanisms described in the next chapters.

4 ADAPTIVE ALLOCATION OF PEER- AND SERVER RESOURCES

This chapter focuses on allocation of peers and servers in peer-assisted VoD. First, we motivate our research question and discuss the related work on server and peer allocation. Subsequently, we provide an analytical model for estimation of instantaneous bandwidth demand of such a system and gain insights into relevant system parameters. We present our adaptive approach that fits into the architecture described in the previous chapter. Next, we propose two different allocation policies: The first policy is inspired by the related work, while the second policy introduces novel concepts. In order to evaluate the performance of proposed policies, we apply segment-level simulations. The experiments analyze the policies' overhead, compare the policies with each other, and, additionally, benchmark them against a perfect static allocation. We show that our policies increase the efficiency if applied to content servers and/or peers managed by the system (such as STBs). The results of the simulation study are additionally verified in a testbed using a real streaming application.

4.1 MOTIVATION

Besides the utilization of upload bandwidth of users, a satisfying streaming quality for users requires the employment of dedicated nodes, either servers or peers that already downloaded a copy of the video in the past and can provide this content to other peers. The upload resources of such distributed *caches* can be allocated to guarantee the desired streaming quality, but must be used sparingly to avoid resource wastage. The latter would result in unnecessary traffic costs in the case of servers and unnecessarily congested upload links in the case of peer caches.

Cache dimensioning in peer-assisted systems is more complex compared to pure server-based systems. The reason is that many peers might be not connected to caches and the prediction of required resources in advance might be not accurate enough. Furthermore, not only the current demand (depending on the current number of downloaders) but also the resource supply of the system is dynamic. Therefore, the questions to be answered in this chapter are:

1. How much bandwidth can be contributed by active downloaders and how much bandwidth must be provided by caches?
2. How can we measure the current bandwidth demand of the system?
3. How many caches should be allocated to provide sufficient streaming quality to the currently watching peers?
4. Which peers should the allocated caches serve?

4.2 RELATED WORK

In this section, we discuss related approaches that deal with the allocation of servers and caches in peer-assisted content distribution in order to improve user experience and/or to reduce servers' load. Recent work on how to deal with temporary undercapacity in p2p systems can be classified into two major categories: (1) server and peer allocation policies; (2) alternative proposals that do not allocate additional resources. In the following we discuss each category in detail.

4.2.1 *Allocation of Servers and Peers*

Concerning the allocation of servers in peer-assisted systems, different proposals have been made, which deal mostly with file sharing in BitTorrent-like systems and live streaming systems [38, 128, 118, 155, 48].

In case of BitTorrent, Das et al. [38] propose that overlay providers deploy special overlay nodes (servers running the BitTorrent protocol) to guarantee minimal download speed to users. They introduce a queuing model to estimate the average download time of each peer depending on the current peer arrival rate. The authors propose that the overlay provider uses this model to (1) determine the number of servers needed to guarantee a target download speed up to a certain peer arrival rate and (2) dynamically adapt the number of servers upon a sudden popularity increase. Note that the approach focuses on the *download speed* as a metric and assumes that the servers run the unmodified BitTorrent protocol. Moreover, the authors do not consider the overhead and do not provide any simulative or real-life results.

Similarly, Rimac et al. [128] analyze dimensioning of servers for in BitTorrent overlays. They use a deterministic fluid model to describe the single swarm dynamics and extend the approach to multiple swarms using a stochastic fluid model. The model enables estimation of the amount of server bandwidth required to satisfy the users' demand with a high probability. Thereby, the download time is the main quality metric and no real-time adaptation is considered. The authors further state that the amount of upload bandwidth contributed by peers is crucial to keep the server traffic low and highlight the benefits of controlled scenarios such as utilization of Set-top Boxes (STBs).

In AntFarm [118] a coordinator server allocates peers and servers to swarms to provide minimal service level, which is again measured in terms of download time. Here, the focus is on the bandwidth allocation among *competing swarms*, for which purpose the authors introduce so-called *response curves*. The latter describe the impact of allocated bandwidth on the aggregated system bandwidth. The coordinator always prefers swarms with a steeper response curve increase. Again the focus on the download time as the main metric and the lack of short-term effects (such as stalling in VoD systems) discern from our requirements.

The discussed approaches focus only on the download speed of peers, which is not constrained by the playback positions and playout buffer states. Therefore, they do not deal with the issue of startup delays and stalling times that are relevant for VoD. Nevertheless, they are an interesting starting point for cache allocation in VoD scenario, so that we propose a policy motivated by these works.

In the area of peer-assisted streaming, Wu, Li, and Zhao present a prediction algorithm to estimate server bandwidth demand for peer-assisted live streaming [155]. The approach predicts the future server bandwidth requirements in a multi-channel streaming overlay using the channel's priority, popularity, and streaming quality. The *streaming quality* metric counts the number of peers that have a buffer count $\geqslant 80\%$. Differently to our work, the authors do not address the playback delay explicitly. Furthermore, unlike in VoD, live streaming does not deal with the issue of prefetching and its interplay with the streaming quality. Nevertheless, the idea of taking into account a metric specific for a streaming application is an important aspect, which is also considered in our work.

Similarly, Niu et al. propose a learning framework to predict the required server bandwidth for peer-assisted live streaming [111]. Their trace-driven simulation study demonstrate that large-scale peer departures and flash-crowd effects have severe impact on the system's performance and quality of prediction.

4.2.2 *Other Approaches*

There is a substantial amount of related work covering various mechanisms for peer-assisted VoD. For example, Annapureddy et al. try to improve streaming quality and reduce server traffic in p2p VoD using advanced network coding, segment scheduling, and peer-matching algorithms to improve the throughput and bandwidth utilization for p2p streaming [10]. The authors argue that a p2p VoD system can provide a "play as you download" experience and can efficiently support a playback rate close to 60% to the peer's upload bandwidth with a small startup delay and no stalling times. However, they do not deal with availability issues and service guarantees, which makes servers or peer caches inevitable in a commercial scenario.

Kumar, Liu, and Ross argue that the ratio of slow and fast users determines the p2p VoD performance [88]. They develop a theoretical model that enables calculating the maximum achievable streaming rate in p2p and distinguish three different operation regions: over-capacity, critical, and under-capacity region, depending on the average peer upload rate. The authors stress the importance of prefetching (called buffering here) and propose admission control and scalable video coding to deal with system's under-capacity. This is different in a peer-assisted scenario, where server and peer caches can be allocated to balance the system performance.

Garbacki et al. [48] derive an entropy-based model for p2p VoD to describe the probability of peers being able to exchange segments, which depends on their playback positions. They show that an excess of upload bandwidth relative to the playback rate and appropriate prefetching techniques can reduce server stress efficiently. Furthermore, they propose to avoid a bad user experience (caused by low upload bandwidth) by using *helper peers*, which donate their bandwidth to increase the download speed of their *friends*. However, the helpers cannot be allocated adaptively and require a pre-existing relationship among peers. Furthermore, the authors do not consider streaming related metrics such as playback delay or startup time.

Huang et al. [66] analyze the impact of prefetching on peer-assisted VoD in surplus and deficit modes, which they define in terms of available and required upload bandwidth. They show that prefetching can significantly reduce the server traffic, especially when the bandwidth demand is close to the supply. Inspired by these results we propose to combine prefetching and adaptive server allocation to balance the supply of upload bandwidth close to the demand.

Carlsson et al. concentrate on the support for newly arrived peers in a p2p VoD system [22]. Such newcomers have typically no segments to upload to other peers, which reduces their upload utilization and bartering possibilities. To avoid too long startup delays for such peers, the servers should support such peers explicitly. The authors propose a modified server-to-peer protocol, which provides the newcomers with rare pieces that should be most interesting for other peers. The work considers the average startup delay and the percentage of late pieces. It states that an increase in server bandwidth has a positive effect on both metrics. However, the authors do not address the issue of adaptive allocation of server (or peer) resources and do not consider the possibility of outliers.

There are more works that deal with various aspects of peer-assisted VoD streaming, such as [32, 28, 160, 98, 106, 19]. For example, Choe et al propose a system called Toast (Torrent Assisted Streaming), which combines server-based streaming with BitTorrent [32]. In Toast, the server is invoked if the desired segments cannot be downloaded from other peers. The authors discuss the benefits of finished downloaders staying in the system but do not discuss the possibility of adaptive allocation of server or peers. Furthermore, they use the server traffic savings as their main metric and do not address any metrics related to the streaming quality.

4.3 ANALYTICAL MODEL

This section provides an analytical model for the transient bandwidth supply and demand estimation in peer-assisted VoD streaming. It offers the insights into the design requirements of cache allocation policies.

Let us assume that there is a set of controllable *caches*, which can be either overlay provider's servers or user-owned STBs controlled by the overlay provider.

- In the first case, we assume that the overlay provider has access to a pool of servers that can be either run by the provider or rented, such as Amazon Elastic Compute Cloud (EC2)[1]. In this scenario the overlay provider pays for the total volume of data uploaded from its servers, while Amazon also charges for usage time. Therefore, the provider is interested to avoid unnecessary uploads if the segments can be served by other peers.
- In the second case (STBs), the bandwidth and hosting costs are paid by the users, but the overlay provider must reward the users for their contribution, e.g. by offering price discounts. An accurate estimation of the number of required STBs to serve the current demand enables switching the remaining devices offline and save their baseline power (in Chapter 6 we discuss the energy awareness in more detail).

In both cases the goal of the overlay provider is to ensure a high streaming quality while minimizing the utilization of caches in terms of upload bandwidth and online time. Uploading too few segments will result in unsatisfied users, while too many segments uploaded by the servers will result in unnecessary high costs.

Our model focuses on distribution of a single video stream, though, the model can be easily extended to multiple streams. We further assume that sufficient caches are available to satisfy the demand gap of peers' assistance (i.e. the difference between bandwidth demand and supply). The model does not cover explicitly the distribution of video copies to caches. In case of server caches, this distribution would happen inside of data centers at much higher speed and lower costs compared to the traffic exchange between servers and peers. On the other hand, peer caches would already possess a previously downloaded copy of the video.

With respect to a distribution swarm of a certain video, we distinguish two types of caches (owning a copy of the considered video). A cache is either *active* if it is currently participating in the swarm and is uploading segments of this video to other peers or *passive*, otherwise.

In order to estimate the bandwidth supply and demand in peer-assisted streaming we use the notation listed in Table 1. As already explained in Section 2.2 prefetching is required to pre-load future segments, and, thus, allows to increase the number of segments possibly interesting for the peer's neighbors. A suitable prefetching factor can differ depending on the utilized protocol. For example, in the G2G protocol (see Section 2.2.3) f should be at least 1.2 to allow smooth video playback [105]. Note that both factors f and g depend on the system design and user behavior and that in general $d_r(l) = \min(d, f \cdot r)$ for all $l \in L$.

Furthermore, the peer upload utilization factor g is the ratio of the average upload speed of peers to their upload bandwidth. Ideally, this value should be close to 1 but peers might not be able to exploit their upload bandwidth due to the lack of segments required by other peers (content bottleneck). In real systems utilization values around 0.8 have been reported [97]. On the other hand, a server can typically utilize its upload bandwidth since it possesses a complete file copy.

We can easily compute the total (upload) *bandwidth demand* of the system as

$$R_{total} = |L| \cdot d_r = |L| \cdot f \cdot r. \tag{4.1}$$

1 http://aws.amazon.com/ec2/ (Accessed: 04.11.2010)

Table 1: Key parameters. We also use the time-dependent notation, e.g. $S(t)$, if appropriate.

S	totally available caches (passive and active)
S'	active caches (subset of S)
u_s	upload bandwidth of cache s (in case of homogeneous caches denoted as u')
L	peers active in distribution swarm (downloaders)
u_l, d_l	upload and download bandwidth of peer l (in case of homogeneous peers also u and d)
r	video bitrate ($r \leqslant d_l$ for all $l \in L$)
f	prefetching factor of the system ($f > 0$)
$d_r = f \cdot r$	required download speed ($r \leqslant d_r \leqslant d$)
g	average peer upload utilization factor ($g \in [0; 1]$)
$\|M\|$	size of a set M

That is, the system must assure that all current downloaders receive the data at the video bitrate speed plus the overhead required for prefetching.

4.3.1 *Homogeneous Upload Resources*

In this case, all peers have the same upload bandwidth, i.e. $u_l = u$ for all l. Furthermore, $u_s = u'$ for all caches. The total available upload bandwidth is then:

$$U_{total} = |L| \cdot g \cdot u + |S'| \cdot u'. \tag{4.2}$$

By comparing the total bandwidth supply and demand we can distinguish the following system states (similar to [65]):

DEFICIT MODE is if $D_{total} < U_{total}$ and, therefore, the streaming quality cannot be assured. Additional caches should be allocated to the system.

BALANCE MODE applies if $D_{total} = U_{total}$ where, ideally, all peers should experience satisfying streaming quality.

SURPLUS MODE is observed if $D_{total} > U_{total}$ so that some cache can be removed from the system.

For an acceptable streaming performance the system must be either in the *balance* or *surplus* mode and, therefore, $U_{total} \geqslant R_{total}$ is required. This results (after inserting of Equation 4.1 and Equation 4.2 and some trivial rewriting) in the following required number of caches:

$$|S'| \geqslant max(|L| \cdot \frac{d_r - g \cdot u}{u_s}, 0), \text{ with } u_s > 0. \tag{4.3}$$

4.3.2 *Heterogeneous Upload Resources*

While Equation 4.3 applies only for homogeneous upload and download capacities, in the heterogeneous case, we obtain the total bandwidth supply as:

$$U_{total} = \sum_{l \in L} u_l \cdot g + \sum_{s \in S'} u_s \tag{4.4}$$

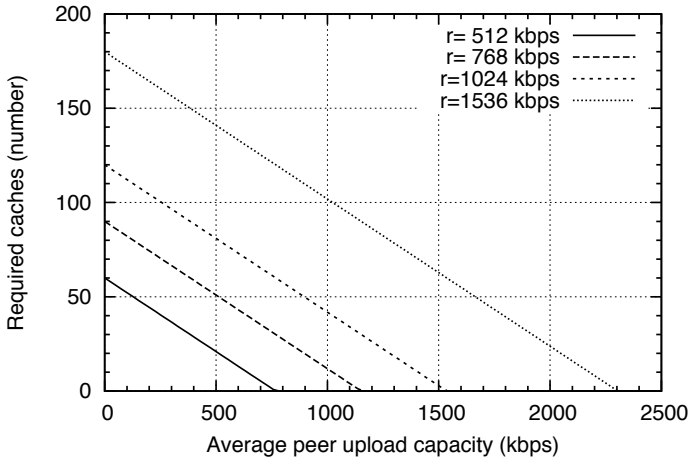


Figure 8: Number of required caches S' for different video bitrates r and peer upload capacities u ($u' = 1024$ kbps, $g = 0.8$, $f = 1.2$, $L = 100$ peers).

and again the bandwidth demand as:

$$R_{total} = \sum_{l \in L} r \cdot f = |L| \cdot f \cdot r. \tag{4.5}$$

Our goal is to find the minimal subset $S' \subset S$ with $U_{total} \geqslant R_{total}$, which requires:

$$\sum_{s' \in S'} u_{s'} \geqslant max(\sum_{l \in L} (d_r - u_l \cdot g), 0). \tag{4.6}$$

The right side expresses the missing upload bandwidth of the downloaders (in case that they cannot provide enough upload bandwidth) while the left side is the total contribution of the selected active caches. Note that the values d_r and g can vary among peers and, therefore, can be only estimated to compute the desired number of caches.

The effect of Equation 4.6 is plotted in Figure 8 with varying peer upload capacity. We can immediately see that in a system with the upload bandwidth of peers being very high (say $u_l = u \geqslant \frac{f \cdot r}{g}$) the cache contribution becomes marginal. For example, with $u = 800$ kbps and $r = 512$ kbps no caches are required, except to assure content availability and to catch up with fluctuations in user demand and upload supply. For a double video playback rate of 1024 kbps the average upload bandwidth of peers must be higher than 1500 kbps to perform well without caches (see Figure 8). Such a case when the peer upload bandwidth is significantly higher than the video bitrate is rather rare for typical Digital Subscriber Line (DSL) and cable access profiles [39]. Therefore, additional bandwidth from caches becomes inevitable.

In summary, the actual demand of server contribution depends on three factors: (1) current number of downloading users L, (2) upload bandwidth of the active users u_l, (3) users ability to utilize the available bandwidth g. The last factor depends strongly on the age of peers, since newcomers hardly have segments to offer, while peers staying in the swarm longer can utilize their bandwidth to a high degree.

Ideally, the caches could contribute just enough upload bandwidth to balance out the available bandwidth in the system by resolving Equation 4.6. However, since the situation can change any time due to departure or arrivals of peers, the overlay provider must allocate cache bandwidth adaptively to avoid serious degradations in streaming quality.

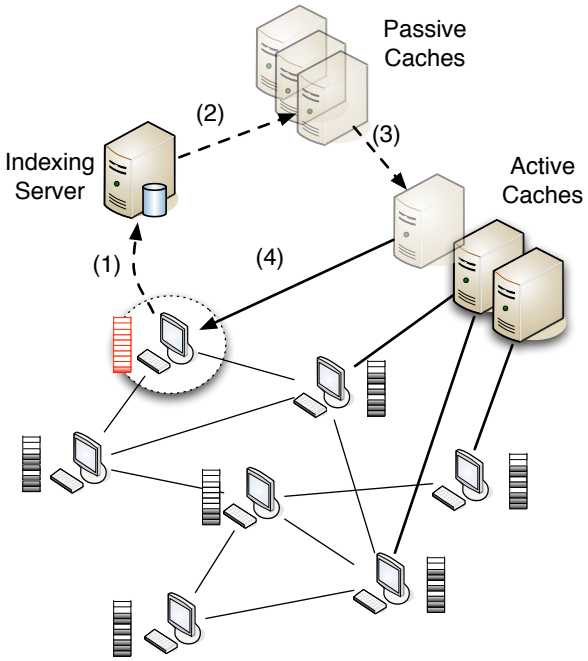


Figure 9: Example of a cache allocation policy applied to a single swarm.

4.4 CACHE ALLOCATION POLICIES

In this section, we first describe the general cache adaptation framework before presenting our policies. The first policy is inspired by the related work, such as [118] but was previously applied only to the BitTorrent swarms. The second policy is tailored to the specific requirements of VoD systems. For the time being, we focus on a *single swarm* workload. However, the realization of our policies (described in Section 4.7) demonstrates that multiple swarms can be supported easily.

In order to provide high quality of service for the users, our system utilizes the architecture presented in Section 3.1 with the modifications depicted in Figure 9. The indexing server monitors the peers' performance (step 1). Based on the peers' reports, the indexing server determines the required upload bandwidth and allocates caches (step 2). Subsequently, these caches join or leave the video swarm (step 3) and provide the missing upload bandwidth to peers in the most efficient manner (step 4).

Since users might abort the playback without watching the whole video, too aggressive prefetching might waste upload bandwidth. Therefore, exploiting the download bandwidth far beyond the video playback rate is not reasonable. Otherwise, fast peers could consume too much bandwidth and leave fast without contributing enough bandwidth in return. In order to avoid such unfair resource utilization, we limit the utilization of download bandwidth to a reasonable threshold, e.g, two or four times the video playback rate.

A cache allocation policy aims to optimize $S' \subset S$ according to the system parameters (as described in Section 4.3): the current number of peers $L(t)$, their upload and download bandwidth u_l and d_l, video bitrate r, and peer upload utilization factor g. These factors could be measured by the indexing server, but might not be up-to-date due to the swarm dynamics. The factor f depends on the utilized streaming protocol and must also be taken into account while determining S'.

We define an actual cache allocation policy as $P = (\Gamma, \Delta, \Lambda)$ with the following components:

MONITORING MECHANISM Γ: This component collects the information about the swarm performance and state of single peers. This is done by the indexing server that is otherwise used for contact management. While the plain contact membership information is normally reported at a scale of several minutes, the streaming quality information must be updated more often to enable fast reaction on performance degradation.

DECISION METRIC Δ: Different metrics can be used to decide when a cache should join or leave the swarm. The candidate list includes the current cache-to-peer ratio, missing upload bandwidth (difference between the total demand and the total available upload bandwidth), current download speed, and buffer states. We consider the (global) average download speed and playout buffer states as the most promising metrics for VoD.

CONNECTION MANAGEMENT Λ: Once a cache decides to leave the swarm it can do so: (1) immediately once the bandwidth excess was detected, (2) after finishing current transfers, or (3) once the connected peers are able to find new neighbors to replace the departing cache. For the join procedure, the question is to which peers a new cache should connect. A possible improvement is also the decision to which peers to allocate the upload bandwidth.

Note that our system is managed by a provider that controls the caches and the client software, similar to the Skype network [13]. Therefore, we do not address the free-riding and the possibility of modified client software. In order to utilize the same architecture with open software and protocols, additional measures must be applied to prevent false reports from the users. For example, the indexing server could log the peer reports and cross-check them to identify malicious peers and ban them from the network.

In general, the monitoring mechanism can be realized in a centralized (indexing server-based), distributed (multiple indexing servers), or decentralized (using a DHT or via gossiping) manner. For the time being, we assume that a single indexing server manages the monitoring per swarm. This allows for the global view on the swarm performance and avoidance of undesired synchronization effects (such as concurrent decisions of several caches to leave the swarm even so only one should be removed). As we demonstrate later on, the overhead incurred through the peer reports for the monitoring functionality can be kept low compared to the actual exchange of the video segments.

In the following we present two concrete policies that instantiate the abstract components.

4.4.1 *Global Speed Policy*

The goal of this policy is to allocate minimal cache resources necessary to balance the global swarm performance. To achieve this, the indexing server monitors the average download speed from all active downloaders, which is reported in regular intervals (e.g. each 10 seconds) as shown in Figure 10, step 1. Based on these reports the indexing server calculates the average *global speed*:

$$G_{measured} = \sum_{l \in L(t)} d'_l(t)/|L(t)|. \tag{4.7}$$

with $d'_l(t)$ being the latest download speed reported by peer l before t.

According to Section 4.3, if $G_{measured} \geqslant f \cdot r$ the system is acting in the surplus mode, otherwise in the deficit mode. When the global speed falls below the desired level, additional caches are allocated to the swarm. However, if the measured speed is too high, some caches are removed from the swarm (see Figure 10, steps 2 and 3).

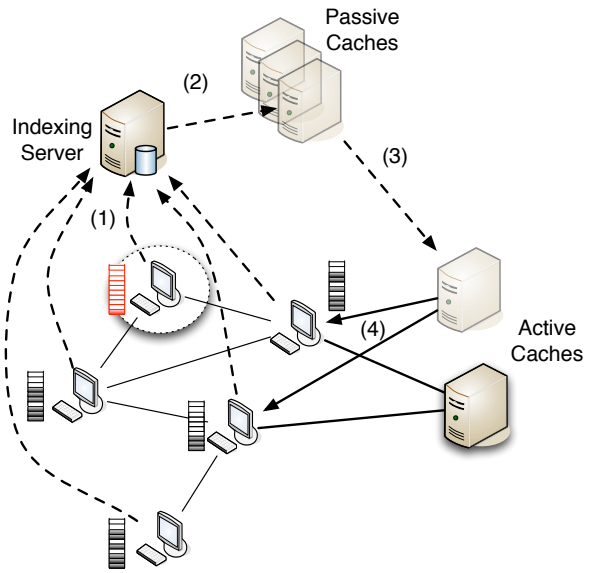


Figure 10: Global speed policy.

Thereby, the target speed is defined accordingly to the desired prefetching factor f' as:

$$G_{target} = r \cdot f'. \tag{4.8}$$

Note that the value of G_{target} is independent from the actual number of the online peers $|L(t)|$. Furthermore, the *intended prefetching factor* f' might differ from the actually achieved prefetching speed. The missing upload bandwidth to be provided by additional caches can then be calculated as:

$$U_{missing} = (G_{target} - G_{measured}) \cdot |L(t)|. \tag{4.9}$$

The variable $U_{missing}$ can also take negative values, in which case there is superfluous bandwidth in the system and the number of caches can be reduced.

The exact algorithm of the global speed based decision metric is presented in Listing 1: The indexing server computes the average global speed $G_{measured}$ over the last interval I (based on the recent peer reports), compares it with the target speed G_{target}, and allocates caches according to their difference (lines 10-17). The indexing server performs these calculations in short periods (e.g. each few seconds) and adds or removes only one cache at once, to avoid too big performance oscillations. Note that there is a special case when no peers are currently present in the swarm. In order to avoid further oscillations in light of minor performance changes, the caches are re-allocated only if the measured speed deviation exceeds 5% of G_{target} (lines 2-6).

Connection management of caches is done similar to the regular peers. A cache that receives an instruction to leave, waits until the transmission of current segments is finished to avoid upload of incomplete segments. Contrary, a join action can be performed immediately.

Reporting Overhead

In order to achieve good accuracy, peers must report their download performance frequently. There are two ways to determine the required report frequency for the current set of active peers $L(t)$:

1. *Fixed reporting interval* I at the client side, which results in the variable number of regular reports at the indexing server. Here, the indexing server must

Listing 1 Decision metric Δ of Global Speed Policy

Require: Pre-defined prefetching factor f', currently active peers $L(t)$, latest peer reports $\{d'_l(t)|l \in L(t)\}$, available caches S, active caches S'
Output: Active supporters S'

```
1:  G_target ← r · f'
2:  if |L(t)| = 0 then {No bandwidth demand}
3:      while |S'| > 1 do {Assure content availability}
4:          s ← random(S')
5:          S' ← S' \ {s} {remove random cache}
6:      end while
7:  else
8:      G_measured ← average(d'_l(t)|l ∈ L(t))
9:      G_diff ← G_measured − G_target
10:     if |G_diff| > 0,05 · G_target then
11:         if G_diff < 0 then
12:             s ← random(S \ S')
13:             S' ← S' ∪ {s} {allocate random cache}
14:         else
15:             s ← random(S')
16:             S' ← S' \ {s} {remove random cache}
17:         end if
18:     end if
19: end if
```

process $|L(t)|/I$ client reports per second, which might become a bottleneck for thousands of videos and ten thousands of users.

2. *Fixed amount* A *of reports at the indexing server* results in variable reporting interval $I = A/|L(t)|$ for the peers. This value must be adjusted each time the number of peers is changing in the swarm and can be sent back by the indexing server upon peers' reports (see Figure 10, step 1).

Figure 11 shows the comparison of both approaches. As we can see the second option (fixed amount of reports) scales better with the increasing number of active downloaders, especially with large swarms.

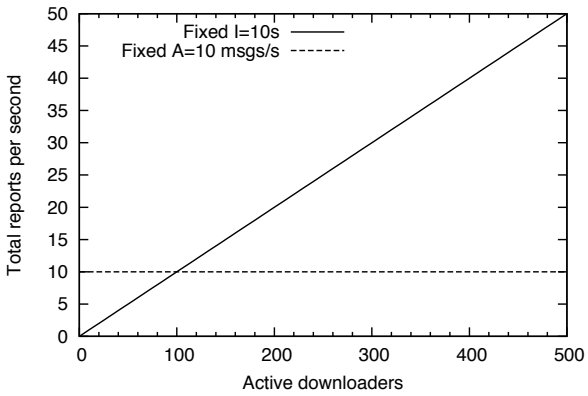


Figure 11: Reporting overhead of the Global Speed policy.

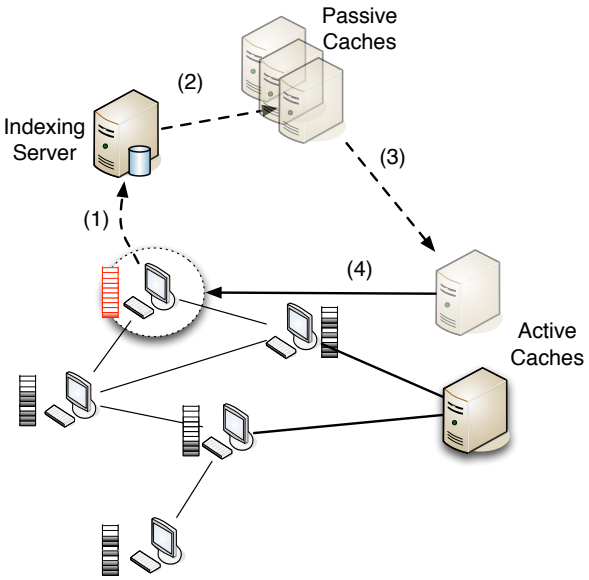


Figure 12: Supporter policy.

4.4.2 *Supporter Policy*

The *supporter policy* differs from the global speed policy by concentrating on the *playback continuity* rather than on the download speed. This policy also addresses another possible drawback of the global speed policy where some peers might experience bad performance even if the global speed is balanced. Therefore, instead of using the average swarm performance as a metric to allocate caches, the supporter policy concentrates on the peers experiencing bad (streaming) performance. In summary, the supporter policy addresses the possible drawbacks of the previous solution in order to avoid:

- bad streaming experience for a subset of users, even if the average performance is high,
- too frequent status reports to the indexing server from too many peers,
- high download speed but stalling playback (due to the lack of desired segments in the peer's neighborhood).

The approach is depicted in Figure 12 where each peer maintains the so-called *high priority set* which is at least as large as the playout buffer (as explained in Section 2.2.2). Instead of periodic download speed reports, a peer sends only the *suffering* status message to the indexing server in the case when the peer cannot download high priority segments fast enough (step 1). Only peers that experience (potential) performance degradation send these messages. Once the indexing server receives a certain amount of such requests in a given time, a passive cache is selected to become a *supporter* (step 2) and joins the swarm (step 3). The supporter receives from the indexing server the list of peers considered as *starving* and serves those peers exclusively (step 4). The supporter stops serving peers that are not considered as starving anymore. Later, if no more starving peers are assigned to a supporter, it can leave the swarm.

A supporter accepts only a reduced number of neighbors, so that it can provide segments at a speed greater or equal to the playback rate. The number of supported neighbors should be not higher than u_s/r so that starving peers can recover fast and smoothly continue with the playback. In order to supply starving peers as fast

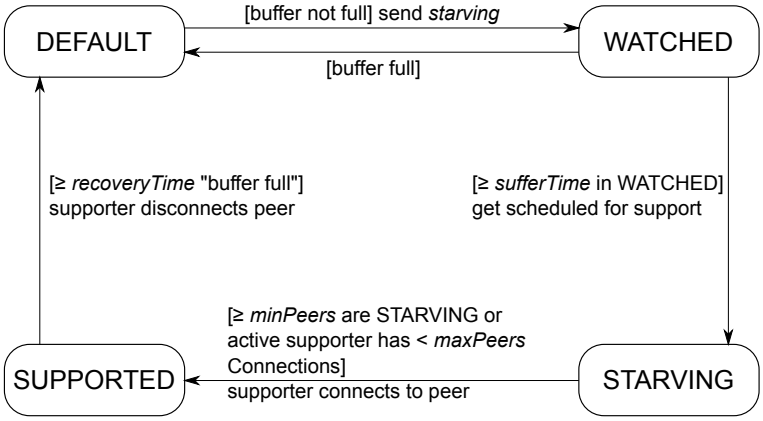


Figure 13: Downloader states in the supporter policy.

as possible, a supporter exclusively serves the peers assigned to it. As long as the supporter has free upload slots, additional suffering peers can be assigned.

Supporter's Decision Metric

An important question is when a peer should be considered as *starving*. A naive approach could consider peers as starving only if their playback is stalling because the segments to be played next are missing in peers' playout buffers. This has two drawbacks: (1) users might experience a stop-and-go behavior of undesired stalling and recovery phases and (2) in the pre-buffering phase the peers might be considered as suffering even if the swarm has sufficient resources available. In the latter case, the pre-buffering phase would be considerably short even without supporter assignment to newcomer.

To avoid these undesired effects, we try to identify *starving* peers before the actual stalling happens while tolerating startup delays of few seconds. This results in the following states of peers monitored by the indexing server (cf. Figure 13):

- *Default*: peer's playout buffer is full.
- *Watched*: peer was missing segments in the playout buffer recently.
- *Starving*: peer time in the watched state reaches the *sufferTime* threshold.
- *Supported*: total number of starving peers reaches the *minStarving* threshold.

The algorithm to decide when and which peers should be considered as starving and receive help from supporters is described as a peer state diagram in Figure 13. As shown there, we start *watching* peers if they miss segments in the playout buffer. A peer that enters the *watched* state sends a suffering report to the indexing server. Since missing segments in the playout buffer can have a transient nature, watched peers are considered as *starving* only if they were not able to fill their buffer in the pre-defined time interval (called *suffer time*) being an important parameter of the mechanism. Only in the starving state a peer will be connected to active supporters that currently serve less than *maxPeers*. If no free supporter can be found, a new supporter is allocated, but only if the number of unserved starving peers reaches the *minStarving* threshold. Note that this condition also covers peers being too long in the startup phase. Directly after joining the video distribution swarm, such newcomers will into the watching state and, if other peers cannot provide them required segments fast enough, they will be served by supporters.

The decision metric calculation in the indexing server is depicted in Listing 2. It uses the collected reports from peers, the knowledge about the current assignment of

Listing 2 Decision metric Δ of the Supporter policy.

Require: Latest reports of all peers P, active and passive supporters S, $minStarving$ threshold
Output: Active/Passive supporters, assignment of starving peers to supporters

1: $P_{recovered} \leftarrow$ Peers that were supported before and recovered
2: $P_{starving} \leftarrow$ Peers that should receive support
3: $S_{available} \leftarrow$ Active supporters with free upload slots
4: **if** $|P_{starving}| - |P_{recovered}| > minStarving$ **then**
5: $\quad m = |P_{starving}| - |P_{recovered}| - \sum_{s \in S_{available}} slots_{free}(s)$ {additional slots required to support starving peers}
6: $\quad$ **while** $m > 0$ and $|S_{passive}| > 0$ **do**
7: $\quad\quad s \leftarrow random(S_{passive})$ {activate additional supporter}
8: $\quad\quad S_{available} \leftarrow S_{available} \cup \{s\}$
9: $\quad\quad S_{passive} \leftarrow S_{available} \setminus \{s\}$
10: $\quad\quad m \leftarrow m - slots(s)$
11: $\quad$ **end while**
12: **end if**
13: **for all** $l \in P_{starving}$ **do**
14: $\quad$ assign l to the next free $s \in S_{available}$
15: **end for**
16: {The deactivation of supporters happens implicitly once all upload slots are empty (see Listing 3).}

peers to supporters, and the *minStarving* threshold. After the calculation of the new peer states in lines 1–3 (according to Figure 13) the decision is made whether new supporters are required (line 4). If they are required and additional slots are needed, additional supporters are activated (lines 6–11). Finally, starving peers are assigned to supporters with free slots (lines 13–15).

Supporter's Connection Management

The algorithm applied by the supporter is shown in Listing 3. The supporter receives from the indexing server two lists: (1) starving peers to support and (2) recovered peers that should not be supported any more. At first, connected peers that left the starving state and do not fall back in the *recover* time interval, are disconnected from the supporter (lines 4–6). Then the supporter connects to new starving peers that were not supported yet (lines 7–9), if sufficient slots are available (lines 10–12). Finally, if all assigned peers have been served and no new peers have been assigned, the supporter leaves the swarm (lines 14–15).

An interesting property of the supporter policy is that we do not need the availability guarantee as required for the global speed policy for swarms with (temporarily) no downloaders (cf. Listing 1, availability condition). A peer that joins an empty swarm will immediately start to send suffering reports to the indexing server that, in turn, will discover that there are no active supporters (and other peers besides the newcomer) and assign a supporter to the newcomer.

Reporting Overhead

Since the peers send reports only if they are not able to fill their playout buffers in time, the reporting overhead depends on the system load. In the surplus mode, only the newcomers will need some time to fill their playout buffers and therefore the load on the indexing servers will be relatively low (it will be proportional to the peer arrival rate). In the balance mode, the load will slightly rise, since some peers might experience temporary difficulties to fill their playout buffer, while in the deficit mode

Listing 3 Connection management Λ of a *supporter*.

Require: Currently connected peers, peers to be supported
Output: Updated list of peers to support

```
1: P_recovered ← Peers that were served by this supporter and recovered
2: P_starving ← Peers that should receive support (sorted by descending waiting
   times)
3: P_connected ← Peers that are connected to this supporter
4: for p ∈ P_recovered do
5:     disconnect(p)
6: end for
7: for p ∈ P_starving \ P_connected do
8:     if slots_free(self) > 0 then
9:         connect-to-and-support(p)
10:    else
11:        return   {No free slots left}
12:    end if
13: end for
14: if slots_utilized(self) = 0 then
15:    leave-swarm(self)
16: end if
```

more peers will experience problems and send *suffering messages* to the indexing server. The latter case is, however, rather improbable in case of well-dimensioned caches.

4.4.3 *Support for Heterogeneous Caches*

So far we did not consider that the resources of caches, especially their upload bandwidth, could vary a lot. For example, in the STB scenario (as described in Chapter 6) the overlay provider controls both low-capacity STBs (*peer caches*) with upload bandwidth in the range from 128 kbps to few Mbps and content servers (*server caches*) with upload bandwidth from several Mbps to one Gigabit per second (Gbps). That means that a server cache can serve much more peers in parallel than a peer cache. Additionally, the costs of traffic served by different caches might differ so that the overlay provider prefers serving segments from peer caches rather than from server caches if possible.

These considerations result in the following additional rules that can be combined with both of the presented allocation policies:

1. If several caches are available for allocation, select peer caches first.
2. If some caches are to be removed from the swarm, select server caches first.
3. Calculate the number of upload slots of cache s as $\mathrm{slots}(s) = u_s/r$ where u_s is the upload bandwidth of s and r is the video bitrate. If $u_s < r$ applies, more than one cache must be allocated or removed at once (to achieve $\sum u_s \geqslant r$).
4. Allocate only caches that possess the desired segments.

Rules 1 and 2 reduce the load on server caches by prioritizing peer caches. Note that in case of the supporter policy rule 2 is applied implicitly because a supporter leaves the swarm if it has no starving peers to support. Rule 3 takes the heterogeneous upload bandwidth of peers into account. Finally, Rule 4 supports peer caches that possess only an incomplete copy of the video. Therefore, as a general rule of thumb, *partial caches* should be allocated only if they possess at least 50% of the video. For the supporter policy we can also reuse the knowledge about playback positions of

starving peers to calculate the desired segments and to select partial caches owning these segments.

4.5 EVALUATION METHODOLOGY

This section describes the methodology applied to evaluate the proposed adaptive allocation policies. The policies should improve the streaming quality and reduce the server traffic compared to static allocation of cache resources. In this regard, we focus on the *server caches* in the first place and use the *perfect* static allocation of server caches as a benchmark (which would require perfect prediction in practice).

Our second goal is the verification, whether the streaming-centric supporter policy can outperform the global speed policy in terms of average and outliers' performance. We further aim to investigate the scalability and overhead of our policies.

Special attention is paid to the utilization of peer caches additionally to server caches and the related impact on streaming performance and server traffic. In the latter case we use not only the static allocation of server caches but also the pure server-based video delivery as a benchmark.

In order to evaluate the performance of the proposed allocation policies, we implemented them in an extended version of the OctoSim simulator [16]. This discrete event-based simulator models data transfers at the level of file segments. As the underlying streaming protocol, we implemented in the simulator the G2G protocol (see Section 2.2.3) according to the specification provided in [105]. All simulation runs were repeated ten times and we provide average values over all ten runs together with their standard deviations unless mentioned otherwise.

4.5.1 *Workload*

The performance evaluation is done using the following workloads (see also Table 2), which provide either the short-term or long-term view on the system performance:

1. **Basic:** constant peer arrival rate with short YouTube-like videos,
2. **Diurnal:** full length movies with the typical 24 hours usage pattern.

We first describe their common setup before covering the specific properties of each workload.

In all scenarios the overlay provider can either allocate cache servers *statically* to a particular video or, alternatively, an adaptive allocation policy can be used to allocate servers on demand.

The peers are divided into three groups based on their upload capacities, representing slow (DSL with 256 kbps), moderate (high-speed DSL or cable with 512 kbps), and fast (Ethernet with 1024 kbps) Internet connections similar to [65]. The relative size of these user groups is 0.3, 0.5, and 0.2, respectively. We also limit the maximum download bandwidth of peers to 2048 kbps to avoid fast peers consuming too much download bandwidth. In any case the download limit is four times the video playback rate and, therefore, sufficient even for an aggressive segment prefetching. Since even for high-end users greedy utilization of the upload bandwidth might result in throttling through the network operator, we also limit the maximum upload bandwidth of Ethernet users to 1024 kbps to avoid too unfair resource utilization. Finally, each server has an upload bandwidth of 2048 kbps.

We model the session length of peers as follows:

$$T_{session} = \min(T_{startup} + T_{playback} + T_{stall}, T_{departure}). \tag{4.10}$$

Here, users might stop the session without watching the complete video after the random time $T_{departure}$. We model the departure process by letting peers on average

Table 2: Evaluation workloads.

Parameter	Basic	Diurnal
Simulation duration	30 minutes	24 hours
Video length	5 minutes	70 minutes
Video bitrate	512 kbps	
Available servers	10	10–40
Server upload bandwidth	2048 kbps	
Peer groups (relative sizes)	0.3, 0.5, 0.2	
Peer upload bandwidth (per group)	256, 512, 1024 kbps	
Peer download bandwidth	2048 kbps	
Arrival rate (exponential)	12 peers/min	100–1100 peers/h
Playout buffer size	10 seconds	
Departure rate	~50% of video	~39% of video

abort the playback and depart from the swarm after 50% (or 39%) of video length after they start playing. The peers are enforced to upload the content as long as they watch it.

Basic workload

The first workload models a VoD scenario where an overlay provider offers short clips, such as trailers, news reports, or YouTube-like user-generated videos, and tries to reduce its costs and increase system scalability by deploying a peer-assisted system. The specific properties of this workload are short session times (~ 2.5 minutes per peer) and the total simulation duration of 30 minutes, which result in the total number of 360 peers. The video length and peer departure rate of this workload are modeled similar to the MSN Video traces presented in [65].

Diurnal workload

VoD systems are used not only for short videos on YouTube-like portals but also for pay-per-view online video rental shops such as Maxdome or Netflix. The offered content are mostly serials and full-length documentary or motion pictures. The increased content length requires also a longer simulation duration (24 hours instead of 30 minutes) and allows for other specific properties of video workloads, such as longer session times.

To evaluate the policy performance in a scenario with full-length movies, we consider a 24-hours workload with the characteristical *diurnal behavior* (see Figure 14). This behavior describes the variable arrival rate of user requests over different periods of time [76]. A typical example is the changing request intensity between the morning and evening hours.

We model such diurnal behavior according to Yu et al. [159] who measured a large VoD system deployed by China Telecom over the period of 219 days. Based on their findings, we model a variable peer arrival rate as shown in Figure 14. We set the video length to 70 minutes and let peers depart from the swarm after watching 39% of the video (randomly distributed). The calculation of the departure rate is again based on [159]. Because of the higher number of concurrent peers (up to 450 peers in the peak hours), the maximum server number is also increased from 10 to 20, 30, or 40 servers.

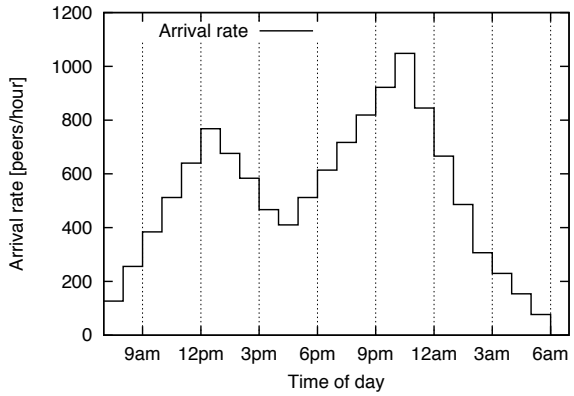


Figure 14: Varying peer arrival rate with the diurnal workload.

This workload is also suitable to evaluate the allocation of *peer caches* since peers that downloaded the content in the beginning of the day can be reused later on to serve new peers. Since the number of peer caches is growing continuously with the time only a subset of them is required later on with the actual number depending on the system dynamics. In this regard, the extended version of allocation policies can be applied to further offload the overlay provider's servers (see Section 4.4.3).

4.5.2 *Metrics*

We use the following playback specific metrics to evaluate the performance of the streaming swarm: *startup delay* until the video can be played and *stalling times* during the playback. Thereby, we apply the startup decision proposed for the G2G protocol: the playback starts after the initial playout buffer (which corresponds to the high priority set) is filled and the remaining download time plus 20% overhead is smaller than the video duration. Whenever appropriate, we also consider the *total playback delay* of peers (or total delay for short), which is simply the sum of startup delay and stalling time per peer. We consider especially the 50th and 95th percentiles of the delays, expressing the maximum delays experienced by 50% or 95% of users, respectively.

At the side of the overlay provider, the main metric is the *hosting* cost of its servers. To this end, we focus on the bandwidth cost incurred by the uploaded data. We further distinguish the amount of data uploaded from the servers and peers to demonstrate the cost savings of the proposed solution.

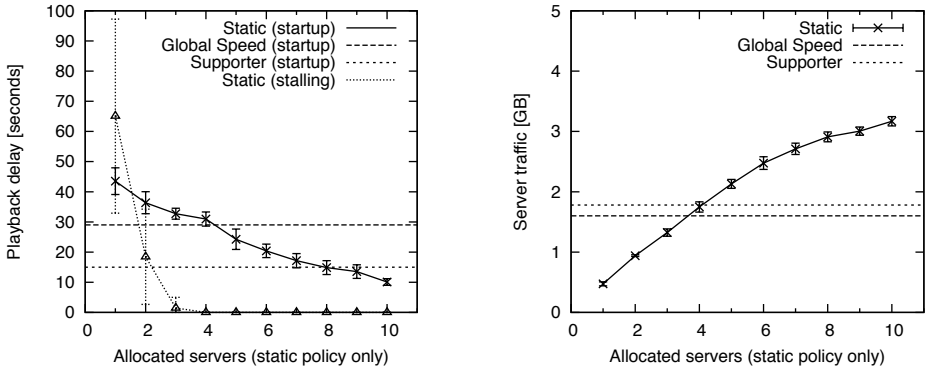
Last but no least, we measure the *reporting overhead* incurred by the adaptive allocation policies by measuring the number of peer reports arriving at the indexing server. In this regard, we assume binary encoding of reported values (average speed for the global speed policy, starving state for the supporter policy) resulting in the message size that easily fits into the standard IP payload size of 1500 bytes, which is is the standard maximum transmission unit of IP over Ethernet [143].

4.6 EVALUATION RESULTS

This section presents the evaluation of the proposed allocation policies, which was performed using simulations and presented in Section 4.6. The verification of the obtained results in a testbed is then shown in Section 4.7.

In order to demonstrate the feasibility and benefits of the proposed policies we conduct several experiments that analyze:

COMPARISON OF STATIC AND ADAPTIVE POLICIES compares the performance and costs of adaptive policies to static server allocation.



(a) Stalling and startup delays (95th percentiles).

(b) Server traffic.

Figure 15: Comparison of static and adaptive policies.

SCALABILITY OF THE SUPPORTER POLICY analyzes the scalability and overhead of the supporter policy.

DIURNAL EFFECTS analyzes the impact of the diurnal traffic pattern.

IMPACT OF HETEROGENEOUS CACHES considers the impact of peer cache utilization (STBs), including a comparison with a pure server-based solution.

We expect our policies to eliminate stalling and to perform at least as good as the static allocation policy with perfect predictions of user behavior. Furthermore, the supporter policy should be able to avoid too many outliers experiencing much worse performance than average users. The utilization of peer caches should offload server caches without decreasing the streaming quality.

4.6.1 *Comparison of Static and Adaptive Policies*

In this subsection, we compare the performance of both adaptive policies and the static allocation of server caches. The analysis is performed using the aforementioned *basic workloads* with the following parameterization of single policies:

For the static policy we let the fixed number of servers stay in the network for the whole duration of the experiment. The number of servers is varied for different runs to determine the best static setup (no stalling and startup delays of less than ten seconds).

The adaptive policies can allocate up to ten servers on demand. For the supporter policy we use the following parameters (based on the sensitivity analysis described in Section A.1.2): *minStarving* = 1, *maxPeers* = 4, *sufferTime* = 5 seconds, and *recoveryTime* = 20 seconds. Finally, for the global policy the target speed factor is set to 1.5 (the impact of various factors is presented in Section A.1.1). Furthermore, the global speed policy uses the *dynamic* reporting interval of 10 messages per second independently from the swarm population.

We present the 95th percentiles of the relevant metrics for different configurations of the static policy and the best obtained adaptive policies in Figure 15. Table 3 additionally provides exact values of 50th and 95th percentiles for all three policies with the best configuration.

The results for the playback delays are reported Figure 15a show startup and stalling times for the static policy, and only startup delays for adaptive policies, since no stalling took place (standard deviations for adaptive policies are reported in Table 3). We make the following observations:

Table 3: Performance comparison of various policies with the *basic workload*. Stalling times are not shown since they are zero for all considered configurations.

Policy	Server traffic [GB]	Startup Delay (Std. Dev.) [s]	
		50%	95%
Best static	1.75	12.2 (1.3)	30.9 (2.6)
Global speed	1.60	10.3 (0.9)	29.0 (2.4)
Supporter	1.78	10.3 (0.3)	15.5 (0.5)

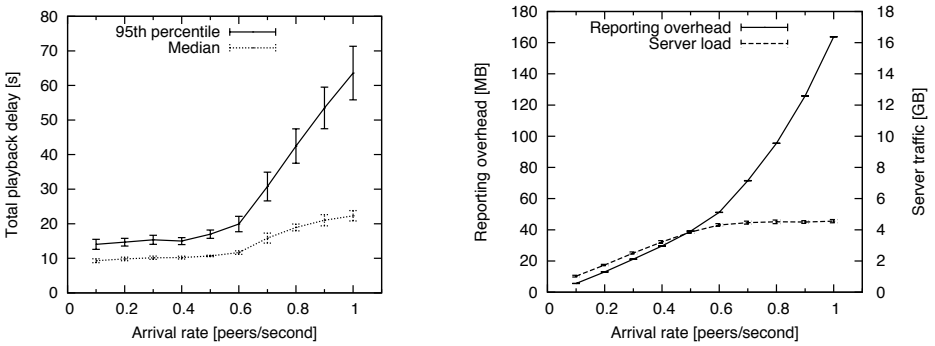
- With the static allocation, the system performs best with 4 servers, though some peers still experience playback stalling. With too few servers the delays are high while with many servers the delays go down due to bandwidth over-provisioning.
- The adaptive policies exhibit comparable performance, while the supporter policy outperforms the global speed policy (50% shorter startup delay). The probable reason is that supporters upload to starving peers in the first place, while the global speed and static policies allocate the bandwidth to random peers.
- In Figure 15b we also show the data volume uploaded by servers. We can see that in the static configuration the server contribution grows almost linearly with the server count. With four static servers they have to upload roughly the same amount of data compared with the adaptive policies. Though the supporter policy uploads slightly more data than the global speed policy, it is able to achieve much lower startup delays (see also Table 3).

In summary, this experiment shows that an adaptive policy enables an overlay provider to reduce costs by allocating available resources according to the swarm performance. This allows to achieve the performance of a well-dimensioned system without knowing the user demand and upload supply in advance. Even if the overlay provider could perfectly estimate the demand for the best static allocation (here with 4 servers), the supporter policy deals better with the system dynamics and provides better streaming performance than the static and global speed policies.

4.6.2 *Scalability of the Supporter Policy*

In this experiment, we analyze the scalability of the supporter policy with regard to the streaming performance, server traffic, and messaging overhead. A proper adaptive policy should be able to allocate less server resources when the system load is low and more servers with the increasing load. We use the basic workload setup (see Section 4.5.1) and vary the peer arrival rate to put the policy under stress. The peer arrival rate is varied in the range $[0.1; 1]$ peers per second and results in the total number of 180 to 1800 peers (the arrival rate of 0.2 peers/second corresponds to the basic setup).

Before presenting the performance results we can calculate the maximum arrival rate that can be supported without streaming quality degradations. According to the analytical model presented in Section 4.3 the total supply of our system can be computed as $U = 537 \cdot L \cdot 0.8 + 10 \cdot 2000$ kbps (537 is the average upload capacity of peers, 0.8 is their average utilization, and there are up to 10 servers, each with 2000 kbps upload bandwidth). Note that the average peer upload bandwidth in our workload is slightly larger than the playback rate (537 vs. 512 kbps), but the higher arrival rate results in a large number of *newcomers*, which have no segments to offer until they fill



(a) Total playback delay.

(b) Content and indexing server traffic.

Figure 16: Impact of the increasing peer arrival rate on the supporter policy.

their playout buffers and prefetch rare segments. Therefore, a higher arrival rate leads to a higher bandwidth deficit and higher demand for servers. The bandwidth demand is $D = L \cdot 1.2 \cdot 512$ (where L is the supported number of concurrent downloaders and 1.2 is the desired prefetching factor.). In order to find the balance mode we set $U = D$ and resolve $L = 10 \cdot 2000/(512 \cdot 1.2 - 537 \cdot 0.8) = 108.2$ concurrent peers. We can apply now the Little's Law [72] expressing the relationship between the mean number of customers in the system L (or downloaders in our case), their arrival rate λ, and their mean time in the system T, as:

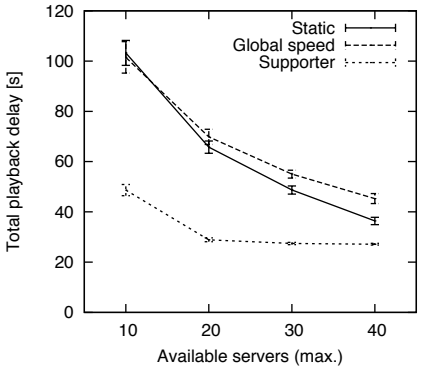
$$L = \lambda \cdot T \tag{4.11}$$

We then insert $L = 108.2\ peers$ that we calculated above. Furthermore, it applies that $T = 0.5 \cdot 5\ minutes$ because an average peer in our setup watches 50% of the 5 minutes long video. Then we can resolve Equation 4.11 for λ in order to obtain that the maximum sustainable peer arrival rate:

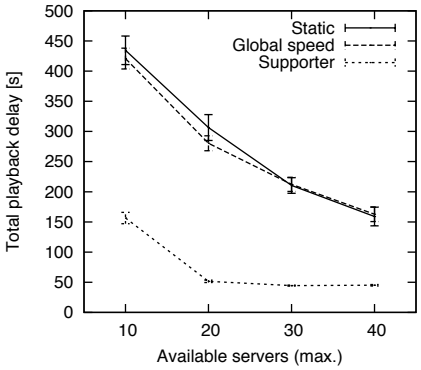
$$\lambda = L/T = 108.2/(0.5 \cdot 5) = 43.3\ peers/min = 0.72\ peers/second$$

The corresponding simulation results for the streaming performance and server traffic are presented in Figure 16. We observe that the median and the 95th percentile of the total playback delay are constantly kept low up to the arrival rate of 0.6 peers per second (see Figure 16a). Above this *saturation point* there are not enough servers to catch up with the missing peer resources and the playback delay starts to increase, while the median increases moderately compared to the 95th percentile. We also observe that the measured saturation point of $0.6\ peers/second$ is quite close to the previously calculated theoretical value of 0.72 (the relative error is $|0.6 - 0.72|/0.72 = 0.16$).

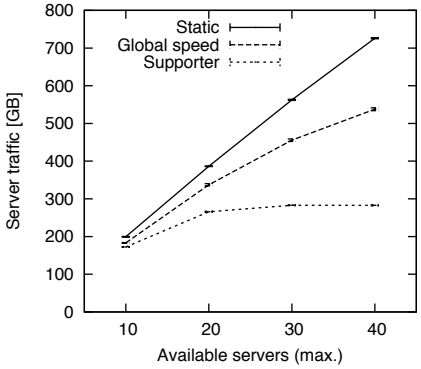
We further observe in Figure 16b that the traffic volume uploaded by servers increases linearly with the peer arrival rate up to the saturation point, after which the load is constant (all servers are allocated). This is consistent with our expectation that the policy scales up to the maximum server capacity. Furthermore, the reporting overhead, which is induced by the *suffering* messages from peers unable to fill their playout buffers, increases linearly with the arrival rate with a steep increase above the saturation point. Below this point the overhead at the indexing server is quite low compared to the content server traffic. For example, with the peer arrival rate of $0.4\ peers/minute$ content servers upload around 4 GB of data while the messaging overhead is only 30 MB, which is less than 1% of the total server traffic.



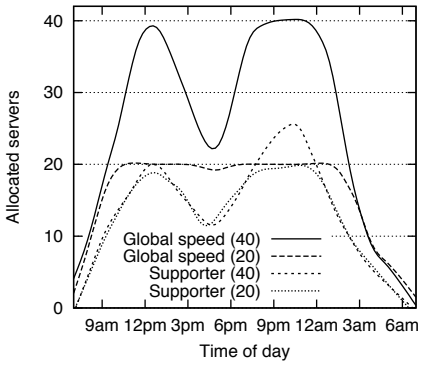
(a) Total playback delay (50th percentiles).



(b) Total playback delay (95th percentiles).



(c) Server traffic.



(d) Server allocation over time.

Figure 17: Performance of allocation policies with the *diurnal workload*.

4.6.3 *Diurnal Effects*

So far we evaluated the short-term behavior of our adaptive allocation policies. For this purpose, the videos were short (and, therefore, also the user sessions and the required simulation duration). In the current experiment we evaluate the performance of the same policies (static, global speed, and supporter) in the much larger *diurnal* scenario (see Table 2). Thereby, we vary the number of available cache servers, while the impact of the available peer caches is covered in the subsequent experiment.

The main results of this experiment are shown in Figure 17. The analyzed metrics are again the total playback delay (both the median and 95th percentile) and the server traffic. We observe the following effects:

- The supporter policy outperforms other policies both with regard to the playback delay (see Figure 17a and Figure 17b) and the server traffic (see Figure 17c). Concerning the median playback delay the supporter policy offers a 58% better performance and with regard to the 95th percentile even an 82% better performance (with only 10 available servers).

- The streaming performance improves with the increasing number of servers where the supporter policy saturates its performance around 25 servers, while both other policies make extensive use of up to 40 servers (see Figure 17d). Nevertheless, the supporter policy performs even with 20 servers better than other policies with 40 available servers. Here, focusing on playout buffers and starving peers provides a significant benefit.

Table 4: Performance of allocation policies with the *diurnal workload*.

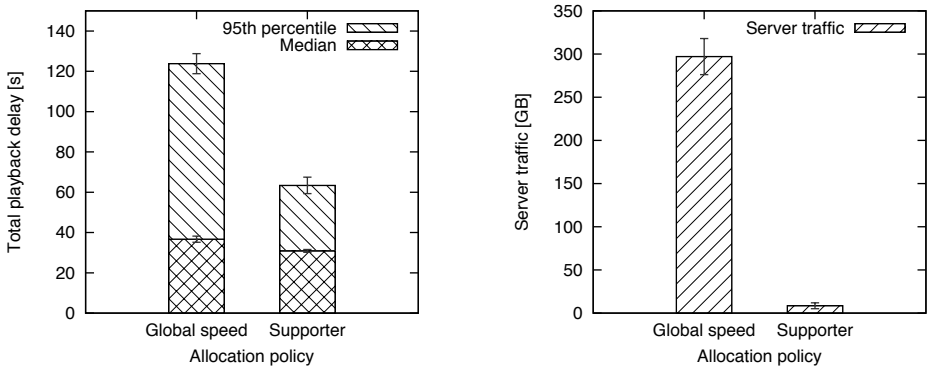
Policy	Allocated servers	Server traffic [GB]	Reporting [GB]	Total delay (Std. dev.) [s]		Stalling [s]	
				95%	50%	95%	50%
Static	30	563	–	210 (13.0)	49 (1.6)	175	14
	40	726	–	159 (15.5)	36 (1.5)	127	5
Global speed	$\leqslant 10$	183	1.16	421 (17.2)	101 (6.3)	372	58
	$\leqslant 20$	337	1.16	280 (12.4)	70 (3.0)	242	33
	$\leqslant 30$	456	1.16	212 (11.7)	55 (1.6)	177	21
	$\leqslant 40$	538	1.16	163 (12.1)	45 (2.0)	127	13
Supporter	$\leqslant 10$	172	2.9	156 (9.4)	49 (2.2)	117	19
	$\leqslant 20$	266	2.9	51 (2.0)	29 (0.8)	25	5
	$\leqslant 30$	283	2.9	44 (0.9)	27 (0.4)	18	4
	$\leqslant 40$	283	2.9	45 (1.2)	27 (0.3)	18	4

- The global speed policy can only slightly increase the streaming performance in comparison with the static allocation. The reason is the less efficient allocation of resources as visible in Figure 17d. It shows that for the maximum number of 20 servers the global speed policy uses all of them during the daytime and reduces its engagement only during the night hours. Contrary, the supporter policy allocates all servers only during the extreme peaks (at 1 and 10 pm) and reduces its engagement significantly around 4 pm and even more during the night hours.
- Comparing the behavior of the supporter policy with 20 and 40 available servers, we also observe that this policy makes only sparingly use of additional 20 servers by allocating some of them during the extreme 10 pm peak (see Figure 17d). This explains why the server traffic increases only moderately if more than 20 servers are available. This is contrary to the global speed policy (and even more aggressive static allocation) which utilizes much more additional servers without a visible streaming performance improvements (see Figure 17c).

Additional performance and overhead results are reported in Table 4. We observe that the reporting overhead is higher for the supporter than for the global speed policy, which is still quite low compared to the server traffic savings. For example, with 30 available servers the server traffic is reduced by $456 - 283 = 173$ GB at the cost of $2.9 - 1.16 = 1.74$ GB of additional reporting overhead over 24 hours, which corresponds to only 21.1 KB/second higher indexing server load.

We also notice that the staling times (both the median and the 95th percentile) decrease with the supporter policy dramatically. This also applies to the stability of the system (visible in the decreasing standard deviation for the supporter policy compared to other policies).

We conclude that the supporter policy again outperforms both the static and global speed based allocation policies. Incurring a relatively low reporting overhead, the server traffic can be drastically reduced up to 47 and 61% compared to the global speed and static policies, respectively. At the same time, the streaming experience is improved with regard to both the total playback delay and stalling times. The improvement is significant both for the median user and the outliers.



(a) Total playback delay.

(b) Server traffic.

Figure 18: Performance of adaptive allocation policies applied to *server and peer caches* (diurnal workload).

4.6.4 *Impact of Heterogeneous Caches*

A significant issue in the workloads evaluated so far is the required availability of server caches being able to satisfy the difference between the bandwidth demand and the bandwidth supply through downloading peers. This requires that the overlay provider either has access to a sufficient amount of server resources (either in its own data centers or rented resources). Another possibility is to reuse the resources of peers that already downloaded the content before and whose upload bandwidth could be reused to serve new peers. For the time being we assume that the devices are online and can re-join the video swarms on demand. This applies especially to the provider controlled systems based on STBs (the online and standby policies for STBs are discussed in detail in Chapter 6).

In this experiment, we (1) evaluate the performance of allocation policies applied to the peer caches and (2) compare the potential savings of the peer-assisted system with and without peer caches relative to the pure server-based solution.

For this purpose, we reuse the diurnal workload and apply the extended version described in Section 4.4.3 to the supporter policy . Additionally to servers, finished peers stay available in the system, and can be reused to increase system's performance.

Figure 18 shows the total playback delay of both adaptive policies and the incurred server traffic. We make two key observations:

1. Both policies achieve very similar median streaming performance but the supporter policy achieves a two times better performance for the outliers (see Figure 18a);

2. The server traffic of the supporter policies is much lower than that of the global speed policy with 8.5 versus 297 GB, which is 95% better (see Figure 18b).

While the first observation is consistent with our previous results, the much lower server traffic can be explained as follows: the upload resources of peer caches are much lower than that of server caches (the average of 537 kbps for peers and 2 Mbps for servers), which results in the fact that *much more caches must be allocated to the system*. Allocating them to random peers results in a suboptimal utilization of their upload resources and, therefore, insufficient global speed. For this reason, the supporter policy performs more efficiently resulting in a better situation both for the users (eliminating outliers) and lower server costs.

Additional details including stalling times and standard deviations are consistent with other results and are provided in Table 5 for the purpose of completeness.

Table 5: Adaptive allocation of peer caches (diurnal workload).

Policy	Allocated servers	Server traffic [GB]	Reporting [GB]	Total delay (Std. dev.) [s]		Stalling [s]	
				95%	50%	95%	50%
Global speed	$\leqslant 20$	297	1.16	124 (5.0)	37 (1.6)	90	7
Supporter	$\leqslant 20$	8.5	3.76	63 (4.1)	31 (0.7)	34	7

Comparison with Client-Server

In Table 6 summarizes our results from the diurnal workload by comparing the savings of various policies with the performance of a pure client-server model. The latter case means that all data is uploaded from servers, for which purpose we repeat the simulation with the peer's upload bandwidth set to zero. For the adaptive allocation, we calculate the combined load on content and indexing servers, and consider both the workload with and without utilization of peer caches.

We observe that with the client-server model, the servers have to stream 2,578 GB of video data. A peer-assisted solution with a well-dimensioned static allocation policy (using 40 servers) saves 71.84% of server bandwidth, while the global speed policy (without peer caches) saves 79,09% of traffic. Furthermore, the utilization of peer caches (STBs) with the global speed policy saves further 9% of traffic. Finally, with the supporter policy the same savings can be achieved even without STB utilization, while the setup with STBs results in savings of 99,52%.

In the basic case, using the supporter policy and peer caches, the servers upload only $\frac{8.5\ \text{GB}}{262.5\ \text{MB}} = 33.15$ video copies (video size is 262.5 MB), while there are more than $12,251$ download requests. Note that the optimal savings would be to stream only a single copy of each video from the servers, which would result in savings of $\frac{2578\ \text{GB}}{262.5\ \text{MB}} = 0.9999$ or 99.9%.

This analysis provides estimation for potential overlay provider's savings for *single swarms*. Therefore, this result cannot be generalized too easily to estimate the total savings of a peer-assisted system. In the whole system there are many videos, some of them very popular and others not. For the less popular videos the number of concurrent viewers might be very low, resulting in a more frequent segment downloads from servers. If a user wants to watch several videos in a sequence, the upload bandwidth must be shared among cached videos, thus, further reducing the total upload

Table 6: Potential traffic savings compared with a server-based system (diurnal workload).

Configuration	Server traffic [GB]	Reporting [GB]	Combined traffic [GB]	Savings [%]
Client-server	2,578	–	2,578	–
Static	726	–	726	71.84
Global speed	538	1.16	539	79.09
Global speed STB	297	1.16	298	88.44
Supporter	283	2.90	286	88.91
Supporter STB	8.5	3.76	12.3	99.52

bandwidth of the system. Furthermore, the storage of peers is limited, which will result in older or less popular videos being removed and less available. Finally, the ability of peers to be online at the right time is desired to serve the content from the peers. Therefore, we need additional analysis and additional mechanisms, as we demonstrate in the following sections.

4.7 VERIFICATION IN A TESTBED

Additionally to the simulative evaluation presented in the previous section the adaptive cache allocation mechanisms were evaluated in a testbed. In order to obtain results comparable with those observed in our simulation study, we decided to implement our mechanisms in the open-source Tribler client [147]. Thereby, we focus on the supporter policy that turned out to be more suitable for a peer-assisted VoD scenario. Since the Tribler client is based on the same p2p streaming protocol as our simulation model similar performance can be expected. However, because of the certain protocol optimization of the client implementation, different levels of detail, and the external effects of the realistic deployment, certain deviations can be expected.

In the following, we describe the most relevant details of the prototypical realization before presenting the obtained results and comparing them with the simulations.

4.7.1 *Implementation in a Prototype*

The basic architecture of the Tribler client provides only one node type: the Tribler client itself. This client can act both as a *regular peer*, a simple server (so-called *initial seed*). The client can further run the internal *tracker*, besides acting as a regular peer or initial seed. A client that does not initiate any downloads acts as a stand-alone tracker. The tracker is responsible for the contact management providing the basic functionality of an indexing server where clients can ask for IP addresses and listen ports of other peers. Furthermore, the tracker provides a system administrator with a simple monitoring functionality.

In our scenario three different entities are built by running the client in one of those modes (regular peer, server, or stand-alone tracker). An implementation of the supporter policy requires the following modifications to each of these entities (shown in Figure 19):

1. `SupporterMonitor` module extends the Tribler's tracker with the monitoring functionalities to provide the functionality of an indexing server. More specifically, it can receive suffering requests from streaming clients calculate their state (according to Figure 13), and dispatch starving peer lists to the supporters.

2. `Client` module is a modified Tribler client that additionally monitors the internal buffer state and sends *suffering* requests to the `SupporterMonitor` module if segments are missed in the playout buffer. Each regular peer that is not acting as a supporter server is extended with this module.

3. `SupporterServer` module is realized by modifying the client's contact management and upload behavior. A supporter server communicates with the supporter monitor to receive the list of allocated regular peers. The supporter connects and uploads to these peers, exclusively.

The following use cases have been identified for supporter monitor (they correspond to the monitor's interface shown in Figure 19):

1. A supporter server joins the overlay by registering at the supporter monitor. The registration information contains the ID of the server, its listen port, and the capacity of the supporter server in terms of the maximum number of supported peers.

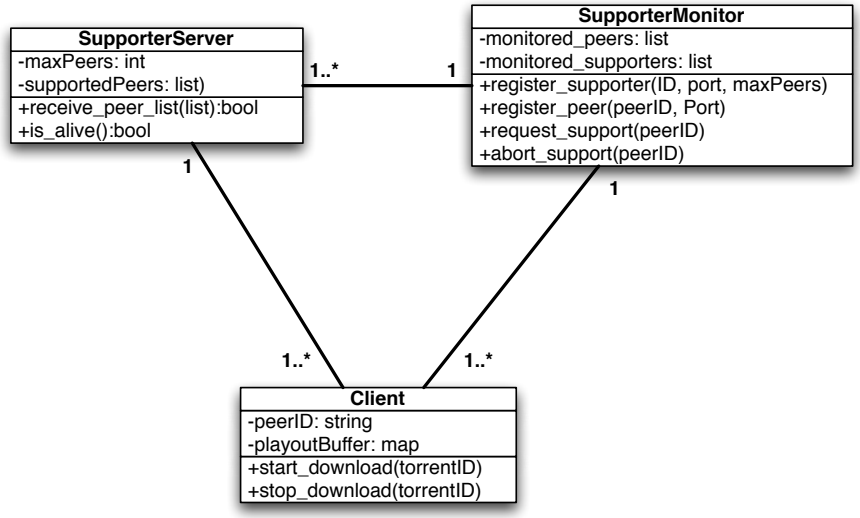


Figure 19: Design overview of the supporter policy implementation (low-level details are omitted).

2. A regular peer joins the overlay and registers at the supporter monitor. For this purpose, the peer signals its compatibility with the supporter-related protocols and sends its ID and listen port. If a peer is member in multiple swarms (one swarm per video), then the peer registers for each swarm separately.

3. A registered peer detects that its buffer is *not full* and reports this to the supporter monitor.

4. A registered peer detects that its buffer is *completely full* and reports this to the supporter monitor.

In order to enable these messaging types, the `SupporterMonitor` extends the HTTP-based communication protocol of the Tribler tracker with additional HTTP resources. The internal algorithm was realized according to the Listing 2. For this purpose, the supporter monitor tracks the calculated states of regular peers and the list of peers allocated to the available servers. After each re-allocation phase the list of peers to be supported are dispatched to the servers using the interfaces offered by the supporter server.

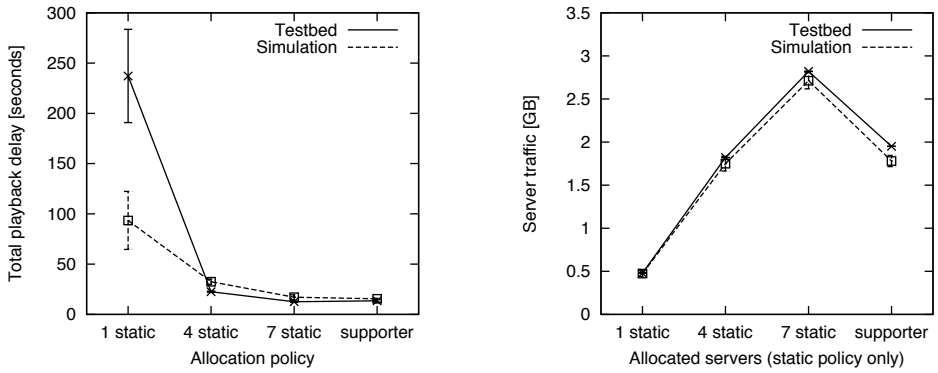
The `SupporterServer` module extends the Tribler client with an additional interface. This interface is based on XML-RPC [153] and provides two methods to control supporter server from the supporter monitor:

1. `receive_peer_list(list_of_peer_ids)`: upon the invocation of this method a supporter server updates its internal list of peers to be supported.

2. `is_alive()`: This method enables the supporter monitor to detect failed or disconnected supporter servers and is invoked in regular intervals.

It is important to ensure that each peer is connected to a supporter. Therefore, the tracker implementation is modified to include at least one supporter in the contact lists provided to peers. We further modify the upload behavior of supporter servers to ensure that only starving peers are served. For this purpose, the relevant `Chocker` class of the Tribler client is altered to serve exclusively the peers whose IDs were included in the last message received from the supporter monitor.

Finally, the regular peer is extended by registering callback handlers that are invoked by the underlying VoD algorithms in regular intervals (once per second). The



(a) Total playback delay (95th percentiles).

(b) Server traffic.

Figure 20: Performance of allocation policies in a testbed.

handler verifies the content of the playout buffer and, if some segments are missing, sends a `request_support` report to the `SupporterMonitor`. The client also sends `abort_support` messages to the `SupporterMonitor` once the playout buffer is full again.

4.7.2 *Comparison with Simulation Results*

The evaluation of the supporter extension was performed in the G-Lab facility [46] that is a national German testbed for the Future Internet and comprises around 150 physical nodes distributed over five German universities. In order to reduce the amount of traffic between the G-Lab sites, each experiment was performed within a single site.

The main goal of the testbed verification is to confirm the results obtained in a simulation. Due to the limited scalability of the available testbed and to enable comparability with the simulations, the setup resembles the basic workload described in Section 4.5.1 as much as possible. Each setup was repeated three times and we report the average values together with the standard deviations. Up to 360 peers participate in a single experiment run.

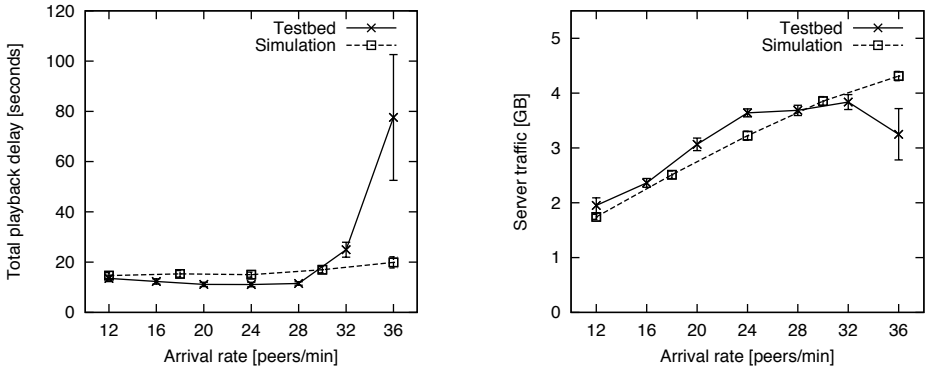
In order to verify the results obtained by simulations we repeated selected simulation setup. In the following, we present two of them: comparison of allocation policies, and scalability of the supporter policy.

Comparison of Allocation Policies

Figure 20 shows the performance comparison for the default configuration of the supporter policy with three selected static configurations. For each setup both the testbed and the simulation results are shown. The testbed results were obtained using the aforementioned setup, while the simulation results are an excerpt of those presented in Section 4.6.1.

The following observations can be made concerning the testbed performance of the policies compared to the simulation results:

- Both evaluation methodologies provide very similar results. The supporter policy provides the same streaming quality as the well-dimensioned 7 static servers (see Figure 20a) while keeping the server traffic at the level of 4 static servers (see Figure 20b). Furthermore, compared with 4 static servers, the streaming quality is much better (14 vs. 23 seconds in the testbed, 16 vs. 33 seconds with simulations).



(a) Total playback delay (95th percentiles).

(b) Server traffic.

Figure 21: Performance of the supporter policy in a testbed.

- The most significant deviations with regard to the playback delays are observed with one static seed (see Figure 20a). There is also a slight deviation with regard to the server traffic with the supporter policy (see Figure 20b). We attribute these deviations to the different level of details of the methodologies in use.

Scalability of the Supporter Policy

Additionally, Figure 21 shows the comparison of testbed and simulation results with regard to the scalability of the supporter policy. The testbed results were obtained similar to the previous experiment but with a variable arrival rate of peers, ranging from 12 to 36 peers per minute. Due to the limitations of the available testbed, it was not possible to repeat this scenario with higher arrival rates. Therefore, the plots include only the corresponding subset of the simulation results from Section 4.6.2.

We observe that for the peer arrival rates up to 32 peers per minute the simulation and testbed results are very similar. This applies both to the playback delay shown in Figure 21a and to the server traffic shown in Figure 21b. Only with the highest peer arrival rate of 36 peers per minute significant deviations occur. We attribute this effect to the link congestion in the testbed when too many peers exchange data simultaneously.

In summary, the performance evaluation with the real client has shown that the results obtained in a testbed verify the results obtained via simulations. That is, the supporter policy outperforms the static policy by providing good streaming quality and limiting the required server's contribution. Furthermore, the scalability of the supporter policy can be verified for scenarios that do not exceed the capacity of G-Lab testbed.

4.8 SUMMARY

In this chapter, we presented adaptive allocation policies for servers and peer caches in peer-assisted VoD streaming. The policies enable the overlay provider to avoid under-capacity and streaming quality degradation in the system. At the same time, they avoid wastage of cache resources. To achieve this, we provided an analytical model that revealed the relevant parameters of peer-assisted VoD. Our approach provides a flexible combination of three components: demand monitoring, cache allocation decisions, and connection management for the caches. We devised two policies: *global speed* that tries to balance the average download speed in the swarm and *supporter* that focuses on the playout buffers of single peers. We showed by means of simulations that adaptive policies can handle unknown user demand and provide high streaming

quality to the users. Furthermore, they can keep the server traffic at a low level. This properties were evaluated with two different workloads (stable peer arrival rate and dynamic diurnal scenario). The results showed that the *supporter* policy was able to outperform other policies (perfect static allocation and *global speed*) by eliminating outliers (visible in 95th percentiles of playback delays). With the diurnal workload this also applies to the median streaming performance *and* the server traffic. We also estimated the potential savings of our system compared to a client-server and STB-based system. In the latter case, the allocation of provider-managed STBs allowed the highest savings of server traffic. Finally, we implemented the *supporter* policy in a real VoD client and measured its performance in a realistic testbed to verify selected simulation results.

In this sense, the chapter provided the basic functionality of a peer-assisted streaming in terms of adaptive peer and server utilization. Due to the scalability issues, accounting for low-level interactions between servers and peers forced us to focus on single videos in the performed evaluation study. In the following chapters, we take the efficient allocation of servers and peers for granted. This allows us to consider distribution of multiple videos in the long-term.

5 INCENTIVE-BASED MANAGEMENT OF OVERLAY TRAFFIC

In the previous chapter, we saw that an overlay provider can significantly reduce its the load on its servers by utilizing peer-assistance. In this chapter, we discuss how network operators can manage the traffic of such delivery overlays without deteriorating their performance. First, we discuss existing traffic management approaches and identify their shortcomings. We derive the requirements for overlay-friendly traffic management, analyze existing approaches with regard to these requirements, and present main insights for an alternative solution. Our contribution is an approach to *indirect* traffic management that aims to meet the requirements of users, overlay providers, and network operators. We evaluate our approach via simulations in the scenario of peer-assisted VoD to show its effectiveness.

5.1 MOTIVATION

Today, pure p2p and peer-assisted overlays are responsible for a large amount of consumer traffic, including the costly inter-domain traffic (see Chapter 1, Figure 1). As we already discussed in Section 2.3, these overlays shift the content distribution costs from overlay providers to network operators. The utilization of peers as data sources introduces a vast amount of content sources, located in various network domains. An overlay routes content from such content sources to content consumers following overlay-specific metrics that are often network-oblivious. This introduces a large amount of costly inter-domain traffic and demand for appropriate traffic management in the network. The situation is different compared to the other prominent type of delivery overlays, namely CDNs, that tend to reduce inter-domain traffic by design [59].

Since the appearance of first p2p-based overlays the research community and network providers were concerned about their impact on the network performance [8, 83, 34, 17, 131]. In the beginning this mostly applied to file sharing overlays, such as Gnutella, eDonkey, and BitTorrent [132]. But with the increasing popularity of video traffic also the relevance of commercial overlays is growing [89]. Therefore, both overlay providers and network operators started to search for appropriate solutions, which manifested in the establishment of the Application-Layer Traffic Optimization (ALTO) group of the Internet Engineering Task Force [134]. Unfortunately, while many of the proposed solutions are able to reduce the costly inter-domain traffic, they might deteriorate overlay performance or introduce additional issues, as discussed below. Before presenting an alternative approach, we give an overview of major traffic management approaches and discuss their shortcomings.

5.2 RELATED WORK

The existing traffic management approaches can be roughly divided into three categories: traffic shaping, network caches, and application layer traffic optimization. While the first two categories describe traffic management approaches applied by a network operator, the latter category includes approaches that are applied either only within the overlay or require a collaboration between the overlay and network.

5.2.1 *Traffic Shaping*

Traffic shaping is one possible technique to reduce the costly inter-domain traffic incurred through p2p overlays [109]. Here, p2p traffic receives lower priority than other types of traffic such as HTTP downloads, which are considered to be more network-friendly. A prominent example is an incident where a network operator, namely Comcast, actively disrupted BitTorrent traffic by resetting connections through spoofed control packets [145]. Such measures, however, typically result in unsatisfied users and overlay providers because of the throttled transfers [109]. Additionally, their impact on network neutrality is highly debatable, since they prioritize certain applications and users [117]. Consistently, network regulation authorities can enforce network operators to change their policies [50].

The first necessary step for traffic shaping is to detect overlay traffic, where different techniques can be used, such as comparing source and target ports of network packets with those used by well-known protocols and applications. After p2p overlays started to use changing ports, Deep Packet Inspection was introduced where not only packet headers but also the application data contained within a packet is analyzed [104]. After the p2p overlays started to encrypt the transferred content, researches proposed to recognize specific patterns to identify p2p traffic [78].

5.2.2 *Network Caches*

Another option are network caches tailored to specific p2p overlays [95, 131, 17, 163, 92]. A network cache is a server deployed by the network operator to store popular content in the local network domain and to redistribute it to local peers. This way the cache prevents local peers from downloading this content from remote peers. Different commercial products exist, such as the Overcache solution offered by Oversi [116]. Caches, however, fail to handle encrypted p2p traffic and the network cache owner might face problems, if caching illegal content.

As described by Lehrieder et al. there are three categories of caches for p2p overlays [92]: transparent caches, ultrapeers, and caches known to the peers. A *transparent cache* is located at the border of a network domain and operates invisible to peers by intercepting requests to peers in remote domains and replying to those requests disguised as the remote peer. Repeated requests for the same content are then served by such a cache to avoid redundant data to cross domain borders. Deployment of transparent caches was evaluated for such p2p overlays as BitTorrent and Gnutella [17, 131].

In contrast to that, *ultrapeers* act like normal peers in the overlay but are managed and inserted into the overlay by the network operator [92]. Their upload bandwidth is significantly higher than that of the other peers, so that they can serve many requests. Furthermore, they cannot be distinguished from ordinary peers by a peer that requests data from them.

The third type comprises caches that are known to the peers of the overlay, i.e. *non-transparent caches*. They are made available to the overlay, so that overlay peers are aware of their origin and might try to request data from them instead of remote peers. Such an approach, therefore, requires a specialized cache discovery [58].

Besides the potential caching of illegal or copyright-restricted content, another serious problem of the discussed caching approaches happens when only one network operator deploys caches: while the local peers can be served very fast from the local caches and receive increased performance, these local peers also become an attractive content source for remote peers, thus, leading to an undesired increase in the inter-domain leaving the local domain [92].

5.2.3 *Locality-aware Overlays*

A different approach is to enhance p2p applications with knowledge about the underlying infrastructure, thus, making them *locality-aware*. A locality-aware overlay modifies peers' internal behavior to exchange data inside of the local network domain first, before contacting remote peers [133, 123]. Different locality-aware techniques were proposed applicable to different types of p2p overlays, such as BitTorrent and p2p streaming [17, 114, 150, 135, 33]. The common idea is to prefer local peers to remote peers. This distinction can be applied, for example, when a peer decides which other peers to connect to or from which peers to request certain data. According to the ISO/OSI model (as described by Zimmermann [161]) these approaches belong to the application layer because they optimize traffic within the applications themselves instead of applying lower level routing and traffic shaping mechanisms applied by network operators.

Such approaches can be further divided into two categories: client-side only traffic optimization and operator-supported traffic optimization. In *client-side only* traffic optimization a peer attempts to determine the locality of other peers through measurements. One such approach is the Ono plug-in for the Azureus BitTorrent client [33]. Ono contacts existing CDNs, such as Akamai or Limelight, to determine the proximity of peers. Before actually transferring data, a CDN redirects users to the providing server that is nearest to the requesting user. An Ono-enabled peer inspects the redirected requests for the hosted content and compares the lists of returned CDN servers with other peers. Peers with a higher overlap in the returned lists are considered as more local.

Other approaches of pure client-side locality use round-trip time measurements to calculate peer proximity [40, 1, 154] or geolocation databases provided by independent parties [80]. A positive aspect of these approaches, however, is the independence from the network operator. On the other hand, these techniques are based on heuristics and might be imprecise or contradict operator's interests.

The inaccuracy of client-side only traffic optimization can be alleviated by deployment of dedicated information services, provided by network operators. Examples of such *operator-supported* approaches are Proactive network Provider Participation for P2p (P4P) [157], BGP-based locality promotion [124], and the "oracle" service [5]. The core idea of these approaches is that the network operator offers special information servers able to provide accurate status information of the network as well as information on peers locations. Whenever an overlay has a choice to select a subset of candidate peers, an information server can be requested to rate the candidates, so that the overlay can decide which connections are more network-friendly. However, the deployment of such a system is costly because a network operator needs to provide these information servers. Furthermore, these approaches rely on a certain trust between overlay users and network operators, since an overlay cannot verify the correctness of operator's ratings.

While different studies confirmed the potential savings of *locality awareness* for network operators, the performance of the overlay is not always improved [17, 114, 124, 135, 157, 79, 73, 84]. The main reason is that locality awareness does not increase the resources of the overlay due to the bandwidth bottlenecks located mostly in the access networks [39]. Piatek et al. state that such approaches may even reduce the robustness of overlays [120].

5.3 REQUIREMENT ANALYSIS

In this section, we present the desired requirements for a traffic management approach that provides a benefit both for the network *and* the overlay. We also discuss how far

the existing approaches from the related work (discussed in the previous section) are able to fulfill them and which lessons we can learn for an alternative solution.

We state the following requirements resulting from the assumptions stated in Section 3.2, the need of coexistence between overlay and underlying network, and further issues discussed in the previous section.

1. *Reduction of inter-domain traffic:* This is a crucial requirements of network operators. A successful fulfillment would reduce the required capacity and installation costs of inter-domain links, as well as the potential payments the lower tier providers pay for transit services. Additional issue here is the differentiation between the inbound and outbound traffic (i.e. entering or leaving the operator's domain), since certain operators pay for them differently [151].

2. *Operator-controlled:* This requirement states that the traffic management mechanism must be applied by network operators. The reason is that operators have the major interest in efficient usage of their infrastructure. On the other hand, overlay-controlled approaches might be never applied unless users or overlay providers see a clear benefit.

3. *Incentive compatibility:* This requirement specifies that overlay providers and users must receive a performance increase. This way the network operator can incite the overlay to cooperate.

4. *Impact:* The traffic management mechanism must address applications that introduce the largest traffic, so that the overall *impact* is visible. It implies that the network operators should focus on file sharing and video streaming applications instead of such applications as e-mail or VoIP [35] (see also Figure 1).

5. *Protocol independence:* This property enables supports for various overlay applications including the future ones. Otherwise the traffic of new overlays cannot be managed unless the support is implemented. This would delay the deployment and increase the implementation costs of the approach. Furthermore, protocol independence facilitates the management of encrypted traffic, in particular in combination with with the next requirement.

6. *Content unawareness:* Network operators must not know which content is distributed by the overlay (or even which users). This has several reasons: Users might be not willing to reveal which content they consume due to privacy issues; Overlay providers might be concerned about the copyright protection of the content; Finally, the network operators want to avoid an involvement in a potential copyright infringement if the content turns out to be illegal.

7. *Network neutrality:* This concept defines how far a network operator is allowed to influence the network traffic of its customers. In its basic notion network neutrality describes the central conflict between efficiency of a system (i.e. operator's network) on one side and fairness of it on the other side [113]. One common interpretation for the Internet is that users can expect all their traffic to be treated equally by the network operators. Tim Berners-Lee, the inventor of the world wide web, defined network neutrality from a QoS perspective as:

   > *"If I pay to connect to the Net with a certain quality of service, and you pay to connect with that or greater quality of service, then we can communicate at that level."*[15]

   Consistently, network operators are not allowed to prefer or discriminate certain applications or overlay providers, if it results in deteriorated QoS [55, 50, 57].

Table 7 summarizes the discussion of existing traffic management approaches presented in Section 5.2 with respect to the stated requirements. The comparison includes

Table 7: Requirement analysis of traffic management approaches.

Requirement	Traffic shaping	Network caches	Locality-awareness	Incentive-based
Traffic reduction (inbound/outbound)	both	inbound only	both	both
Operator-controlled	yes	yes	no	yes
Incentive compatibility	no	yes	no	yes
Impact	yes	yes	yes	yes
Protocol independence	limited	no	yes	yes
Content unawareness	yes	no	yes	yes
Network neutrality	no	yes	yes	yes

three major approaches from the related work and our incentive-based approach presented and analyzed in the rest of this chapter.

Furthermore, traffic shaping enables reduction in the inter-domain traffic, at least in cases when the traffic and application type can be identified reliably. The major drawback however is that this approach clearly violates the principle of network neutrality. Throttling p2p traffic most likely lowers the experienced QoS for the p2p users. Even worse, the active disruption of certain protocols (such as the handling of the BitTorrent protocol by Comcast) might render the p2p-based application completely useless. This discrimination against p2p applications may lead to disgruntled customers of the network operator [152]. As those customers may decide to switch to other network operators, this form of traffic management might even be harmful to the network operators themselves. Another important issue is that the decrease of the overlay performance might seriously impact the commercial success of an overlay provider. For example, in a peer-assisted system a reduction of traffic served from remote peers might require more traffic to be served by overlay provider's servers, which eliminates the benefit of peer-assistance and ruins the business model.

Network caches are able to reduce *inbound* inter-domain traffic for supported protocols.[1] but suffer from certain problems First of all, such caches require a significant investment as the huge data amount being transferred requires large storage space at the caches to serve a lot of requests [53]. Another problem of caches are legal aspects because of unavoidable content awareness. Thus, a network operator might violate the law by storing certain data on servers within its domain. On the one hand, the stored data may be illegal in its own right (e.g., pirated copies of movies or music). On the other hand, there may exist copyright laws that allow the transfer of that data to certain geographical regions only or for a certain amount of time. The network operator that caches such data would have to guarantee that these policies can be ensured [92]. Further limitations of caches are the required investments to support different protocols and their inability to reduce the outbound inter-domain traffic.

Finally, locality awareness can avoid many problems of traffic shaping and network caches but relies on external services and therefore requires a certain level of trust. This risk of wrong locality advises might be too high for the overlay provider and users, who cannot expect significant performance improvement in return [120]. For this reason, network operators deploying the required information services cannot expect *strategic* users to obey their advise and might hesitate to deploy such information services.

1 Unless the caches are misconfigured or equipped with insufficient resources.

Finally, our incentive-based approach was designed to fulfill all of the presented requirements. Thereby, it does not have to be used exclusively but can effectively coexist with the locality awareness in the overlay.

5.4 APPROACH OVERVIEW

The previous sections revealed that current approaches to the management of overlay traffic have not been able to sufficiently solve ongoing problems. The reason is that p2p traffic inherently generates additional traffic within a network and across network boundaries. So far, none of the presented approaches has achieved a solution that is acceptable for all three parties: network operators, overlay providers and users of a p2p application. In this regard, any strategy for traffic management that explicitly throttles inter-domain traffic results in degradation of the overlay performance. This, in turn, results in either poor QoS of the application from user's perspective or in an increased load (and therefore costs) on the overlay providers' servers that have to provide the missing bandwidth.

In particular, client-side locality awareness brings no benefit to the overlay and, therefore, lacks an incentive to be widely adopted by overlay providers and users. The reason is that the performance of p2p overlays for content distribution (such as file sharing and streaming) relies strongly on two factors: availability of the content and the upload bandwidth of participating peers. While the overlay performance only considers these factors globally, the network operators should attempt to improve them in the local domain. Only then they can successfully promote locality without hurting overlay's performance.

We conclude that the attempts to promote locality without first improving content availability and locally available bandwidth might lead to overall deterioration of the overlay performance. In order to overcome such limitations, we propose that network operators provide a clear and measurable *incentive* for locality-aware behavior at the overlay level. To this end, the network operators should promote users that are compliant with overlay and network requirements by increasing their upload and download bandwidth. This approach is able to increase the overlay performance *and* to reduce the amount of inter-domain traffic resulting in the so-called *triple-win situation*:

1. Network operators can reduce the costly inter-domain traffic generated by p2p overlays. At the same time they do not risk to disgruntle their clients.

2. Overlay providers benefit from the increased bandwidth of overlay peers. Their own resources are relieved from traffic load even more than without any active traffic management. Thus, the overlay provider can choose either to save server hosting costs, to increase QoS for the delivered content, or even to increase the video resolution.

3. Users can directly and indirectly benefit from incentive-based traffic management by the network operator. The provision of additional resources directly improves the performance of the overlay and, therefore, indirectly the experienced quality of each overlay user. Furthermore, the users selected for promotion can benefit from free resources without extra fees by using network-friendly applications (or application configurations).

5.4.1 *Incentives*

The key factor for the performance of a p2p overlay (either pure or peer-assisted) is the *availability of bandwidth at peers*. Considering the example of a peer-assisted VoD streaming application, the "p2p effect" is getting stronger, if the total upload capacity of peers can satisfy the total download rate demand. The required download rate here

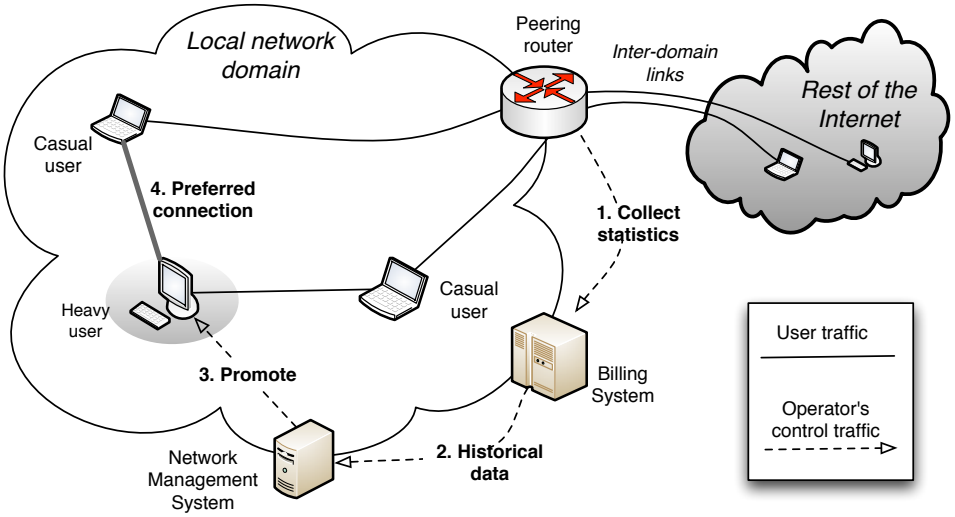


Figure 22: Example of Highly Active Peer promotion from the perspective of local network operator.

must be at least as high as the video playback rate to achieve the "watch-while-you-download" behavior. This is not a trivial task, since the upload bandwidth of peers is typically sparse (e.g., 1:4 or even 1:8 for DSL connections) and users tend to leave the overlay once they finish the download. Thus, providing additional *upload* bandwidth to users makes them better contributors, improves their standing in the system, and increases the overall capacity of the system.

The measures how the available bandwidth can be increased in the local domain (without introducing additional entities, such as network caches) comprise two alternatives: either the upload bandwidth of local peers should be increased or the local peers should stay longer online, even if they do not consume any traffic. While the local operator can accomplish the first part by promoting the users' access profiles, the second part requires the operator to promote only certain users, thus, rewarding only the biggest contributors. Furthermore, by promoting only a subset of peers, the operator can also decide to reward locality-aware peers, thus, providing a clear incentive for this behavior.

Consequently, the proposed mechanism suggests that the network operator increases the total available upload bandwidth in its domain by increasing upload bandwidth for the selected subset of overlay users, promoting them to so-called Highly Active Peers. These users are most likely to increase the overlay performance and to promote locality for the benefit of the network operator. Thus, the mechanism focuses on promoting the best overlay users to HAPs by significantly increasing their bandwidth for sufficient time intervals. Ideally peers promoted to HAPs should also behave locality-aware by exchanging data mostly with local peers. Otherwise, they might use the increased bandwidth to upload data to remote peers and increase the outbound traffic. To enable the re-evaluation of promoted peers, the promotion, and any associated benefits, stays active until the next evaluation phase takes place. The peers that already have been HAPs in the previous phase either remain HAPs (if they have behaved as expected) or are demoted to normal peers otherwise.

The resulting situation is shown in Figure 22. In the local network domain the network operator monitors the usage profiles of peers to identify those having the highest (potential) impact on the overall upload traffic (step 1). The network operator manages this data in its billing and provisioning system that aggregates the historical data and offers it to the network management system (step 2). The network management sys-

tem analyzes this data, selects certain peers and assigns them additional bandwidth, which makes them more attractive for other peers (step 3). Now other local peers should detect these highly capable peers and download content from them (step 4). Additional measures can be applied to assure that other local peers easily discover the promoted peers, and that remote peers do not overuse their resources. While the first measure assures that the inbound traffic is reduced, the latter case prevents a parallel increase in outbound traffic.

5.5 DETAILS

Previous section presented the general idea of incentive-based traffic management through HAP promotion. While this approach provides a clear benefit to the overlay by increasing peers' capacities for free, certain questions arise with regard to its effectiveness for the network operator:

1. How can the promotion be performed in suitable time intervals in a real network?
2. Which users should be promoted, i.e. which selection metrics should be applied to select HAPs?
3. Can we incite the overlay provider or single users to behave network-friendly by promoting mostly locality-aware peers?
4. How to avoid that the increased network capacity is abused by the overlay, which results in an increased inter-domain traffic when the increased bandwidth is used to upload data to remote peers?
5. Is it beneficial for a single network operator to deploy HAP promotion?
6. Which is the impact on the traffic composition, that is, which is the effect on inbound and outbound traffic of network operators?

In this section we describe in detail the concrete mechanisms that give answer to these questions and fulfill the requirements stated in Section 5.3. For this purpose we split the proposed approach into the following steps:

MONITORING MECHANISM: The network operator monitors the performance of local overlay users and collects their usage statistics by the means described in Section 5.5.1.

HAP SELECTION STRATEGY: Based on the monitoring statistics aggregated at regular intervals, network operator calculates a rating metric and decides which peers to offer the possibility to act as HAPs (see Section 5.5.2).

PROMOTION FUNCTIONALITY: The network operator uses automated reconfiguration capabilities of its network equipment to promote selected peers by increasing their upload bandwidth (see Section 5.5.3 for details). This promotion can also include a certain increase in download bandwidth, so that peers immediately see the promotion and understand that their network-friendly behavior was rewarded.

Another important aspect is the dimensioning of the mechanism with respect to the total bandwidth used for HAP promotion, the number of peers to promote, and the additional bandwidth per peer. We discuss them in Section 5.5.4.

The resulting HAP promotion algorithm is executed at the beginning of each promotion interval. Based on the computed rating values, the network operator performs the promotion of peers using the selected rating metric. Promotion is only active for

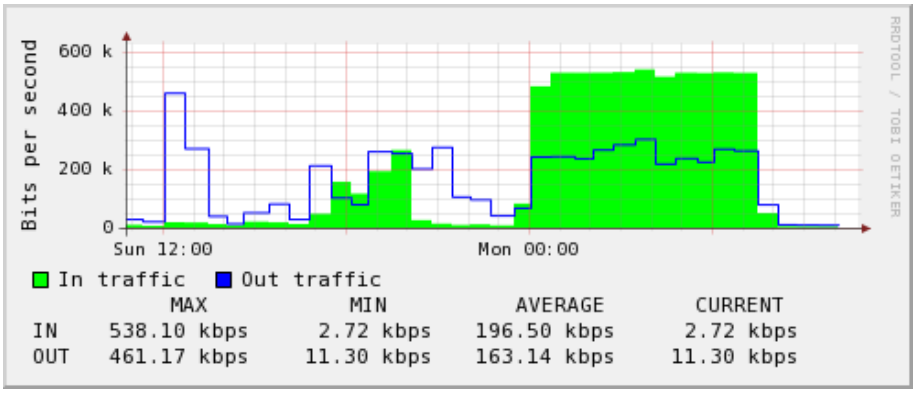


Figure 23: Example of a traffic monitoring of a single user over 24 hours.

the duration of the *promotion interval* τ, since during the next interval a new set of HAPs is selected, while peers not eligible to be HAPs again are reverted back to their normal access profile.

5.5.1 *Monitoring Peers' Behavior*

In order to decide which peers should be promoted to HAPs, a network operator must collect information about their usage profiles. This can be done either through the monitoring functionality of the network operator's network equipment (such as NetFlow [137]), or from the overlay application itself. Both approaches have their advantages and disadvantages, some of which depend on the specific equipment in use by the network operator.

The first option, *in-network monitoring*, makes the mechanism protocol-independent, as it only requires deployment on the network operator side and no extension in the application itself. In order to achieve best results, the network operator should be able to classify the exchanged data according to the protocol type: p2p, HTTP, etc. This can be done by the means of Deep Packet Inspection (DPI) or traffic pattern detection [78]. Otherwise, the collected statistics will contain the data about the total traffic of the given network operator's customer, resulting in a certain deviation. This can be tolerated to some extent by focusing on users generating significant upload traffic volume, since most non-p2p applications are rather making use of upload than download bandwidth.

The second option, *collecting behavior information from the overlay*, requires additional cooperation between the overlay provider and the network operator. For this purpose, the network operator must offer an interface for local peer to report the amount of traffic exchanged with other peers. Unfortunately, this information cannot be fully relied upon as it can be potentially tampered and mislead the network operator into promoting wrong users. This issue can be partially addressed by cross checking the provided information, which, however, complicates the procedure, since some peers could try to collude [81].

Based on this analysis we propose to collect data from the network equipment in the first place. The non-forgeability of this data is its major advantage. Furthermore, this approach does not require any changes to the monitored overlay applications and, therefore, supports any p2p overlay. A real-life example from a network operator monitoring facility is shown in Figure 23. The plot is generated using RRDtool[2] and visualizes the total upload and download traffic of a single peer. Especially, the

2 http://www.mrtg.org/rrdtool/ (Accessed: 15.10.2010)

Table 8: Traffic statistics collected per user and measurement interval t.

Parameter	Description
$V_{up}(t)$	Traffic uploaded
$V_{up}^{loc}(t)$	Traffic uploaded within the local domain
$u(t)$	Upload bandwidth during last promotion interval t
$V_{down}(t)$	Traffic downloaded
$V_{up}^{loc}(t)$	Traffic downloaded within the local domain

active link usage during the morning hours can be attributed to a p2p (file sharing) application.

The per-peer statistics required for our algorithm are listed in Table 8. All of them can be collected either protocol-independent or classified to a certain class of protocols, that are p2p protocols in our case. This ensures two important properties of our solution: *protocol-independence* and *content-unawareness*. The first-level or gateway routers can collect all of the presented parameters, with the upload bandwidth of peers being the only exception, since it is managed by the network management system anyway. Notably, the statics do not have to be available in real time because the approach requires this data at the level of several hours. For example, the network operator can collect them on the daily basis and access them during the night, when the network equipment is less loaded than during the day.

We can justify the assumption that these values are available to the network operator. Since network operators are able to perform volume-based charging for their users, they have to know their upload and download traffic as demonstrated in Figure 23. Furthermore, network operators have a global view on the locality of Internet addresses that belong to their domain, which allows them to distinguish between local and remote traffic. It is also important that monitored values, do not contain specific information about the content of the exchanged data, thus, fulfilling the *content-unawareness* requirement (see Section 5.3). Therefore, the monitoring mechanism does not impose privacy or trust issues between a network operator and its users.

5.5.2 *Selection Strategy*

The goal of the HAP selection is to identify most active and available peers. These peers must be able to act as locality-promoting HAPs and bias the overlay traffic for more locality. The questions to be answered is therefore: which peers should be promoted to HAPs by increasing their bandwidth.

In order to answer this question we consider the relevant properties that qualify a peer to become HAP . In the first place, a peer must be an attractive source for other peers, that is, it must possess popular content in its local cache. Furthermore, this peer should be more available than other peers, which is the case when a peer either consumes more content (becoming a *heavy users*) or keeps the application running in the background without using it (being an *altruistic user*). The latter behavior is also called *seeding*. These both properties do not have to be aligned, since a peer might consume a lot but not seed after finishing its downloads.

Finally, the peer might behave differently with regard to its network operator by either being *locality-aware* or not (see also Section 3.4.4). For example, in BitTorrent networks there is a possibility to install additional plugins that try to bias the client behavior for more locality and, implicitly, more network operator friendliness [33]. Such peers are also good candidates to be promoted to HAPs, either to utilize their

locality-aware behavior or to reward them, thus, inciting other users to change their behavior towards more locality in order to increase the probability to be promoted.

Basic Metrics

Based on the above considerations, HAP selection strategy needs to consider both the relative contribution of the peers to the overlay network as a whole as well as the contribution to the local domain. Thus, we can address the interests of network operator and overlay providers, which are not always aligned. As peers can have rather diverse configurations, all evaluations have to be done relative to the maximum possible contribution in any given period of time, thus, keeping the selection fair to all potential HAPs. In the following we present selection *metrics* suitable to detect HAP candidates according to these criteria. For this purpose, we assume that the network operator has access to the per-user data that describes peer's contribution to the overlay in global and local domains (see Table 8). For simplicity, we set the duration of the measurement interval τ equal to the promotion duration, while in a real setup it depends on the minimal possible possible promotion interval and monitoring equipment capabilities (one day or less appears feasible). It must be also taken into account that too short measurement intervals might lead to wrong estimation of peer's suitability since random or diurnal effects can distort the measured values.

A HAP selection metric assigns a rating value $R(t)$ to each peer p for the currently examined period t, so that the network operator can select the highest ranked peers to be promoted. The basic considered metrics are:

- The *Contribution* metric $C(t)$ represents the total contribution of the peer to the overlay. Peer's contribution is crucial for the overlay's performance, since peers who already contributed a lot, could potentially contribute even more. The metric is calculated as:
$$C(t) = \frac{V_{up}(t)}{u(t) \cdot \tau}$$
with $u(t) \neq 0$ and $\tau > 0$. Note that to keep this metric fair for all customers and agnostic to actual customer's bandwidth the upload volume is divided by its upper bound $u(t) \cdot \tau$ that would have been observed if a peer continuously uploaded data at full speed. Furthermore, this normalization allows for fair comparison between HAPs and normal peers.

- The *Network-friendliness* metric $F(t)$ prefers peers that probably apply locality-aware peer selection or that are popular inside the domain, being more quickly discovered by other local peers. We compute $F(t)$ as:
$$F(t) = \begin{cases} \frac{V^{loc}_{up}(t)+V^{loc}_{down}(t)}{V_{up}(t)+V_{down}(t)} & \text{if} \quad V_{up}(t)+V_{down}(t) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

- The *Seeding Ratio* metric $S(t)$ is the amount of a peer's upload traffic relative to the total traffic volume. It enables selecting peers that are altruistic in the sense that they stay online to provide content and not only to consume it. This allows us to avoid the measurement of actual online time of peers, that is practically rather difficult. We calculate seeding ratio $S(t)$ as:[3]
$$S(t) = \begin{cases} S(t) = \frac{V_{up}(t)}{V_{up}(t)+V_{down}(t)} & \text{if} \quad V_{up}(t)+V_{down}(t) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

3 Note that our definition differs from the seeding ratio common in the BitTorrent networks which is defined as $\frac{V_{up}(t)}{V_{down}(t)}$. Our version has the benefit of normalization, since all values are in the range $[0:1]$. Nevertheless, the resulting order of peers is the same.

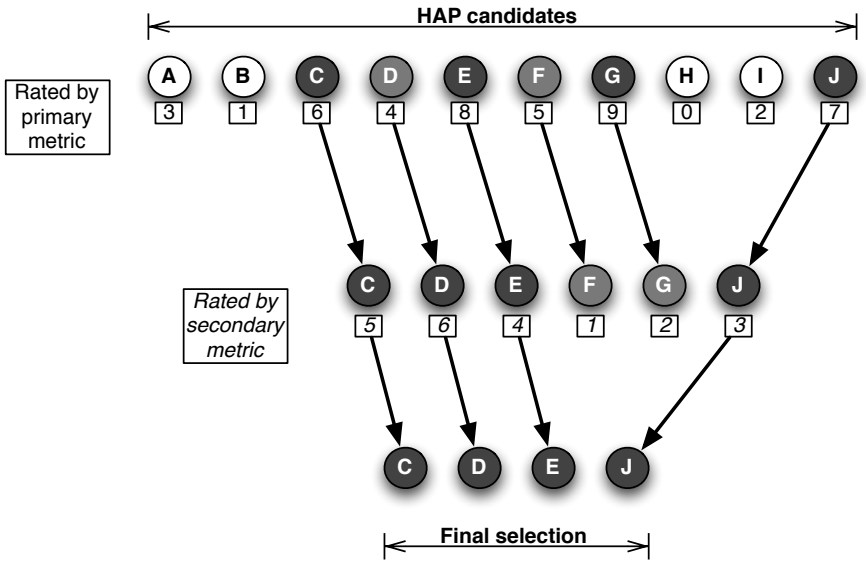


Figure 24: Combination of two HAP selection metrics, 4 out of 10 peers should be selected. 6 peers are selected by the primary metric and then re-ranked by the secondary metric (the numbers in squares denote values assigned by metrics).

Furthermore, we include a *random* HAP selection metric that assigns a random rating value to a peer. It serves as a baseline for comparison since it does not rely on any knowledge about the peers' behavior.

Combining Metric

Each of the previously presented basic metrics can be used either by itself or in combination with other metrics. For example, a combination becomes relevant, if we want to combine the detection of locality-aware and most active peers.

A simple combination possibility is the weighted sum of single metrics similar to the proposal in [135]:

$$R(t) = w_c \cdot C(t) + w_f \cdot F(t) + w_s \cdot S(t). \tag{5.1}$$

Since each metric returns values in the range $[0:1]$ and $w_c + w_f + w_s = 1$ it applies that $R(t) \in [0:1]$, too. Furthermore, by setting any two of three weights to zero, we obtain a single metric. An issue with this approach is the selection of appropriate weights w_c, w_f, and w_s. Because the metrics have different semantics, there is no intuitive way to define them.

Therefore, we propose an alternative way to combine two metrics, such as Network-friendliness and Contribution. The idea is to use one metric as a *filter* before applying the other one. Let us consider a set P of candidate peers N and the primary metric C. We define $C_{r_1}(P)$ as the r_1-th percentile of P according to C. In the first filtering step, we select the best $r_1 \cdot N$ peers as:

$$P_C = \{p \in P | C(p) \geqslant C_{r_1}(P)\}.$$

This step removes all peers that do not show significant activity in the overlay. In the second filtering step, the best $r_2 \cdot |P_C|$ peers are selected resulting in the overall number of $r_1 \cdot r_2 \cdot N$ selected peers. Using Network-friendliness as the secondary metric (and its r_2-th percentile) we obtain:

$$P_{C,F} = \{p \in P_C | F(p) \geqslant F_{r_2}(P_C)\}.$$

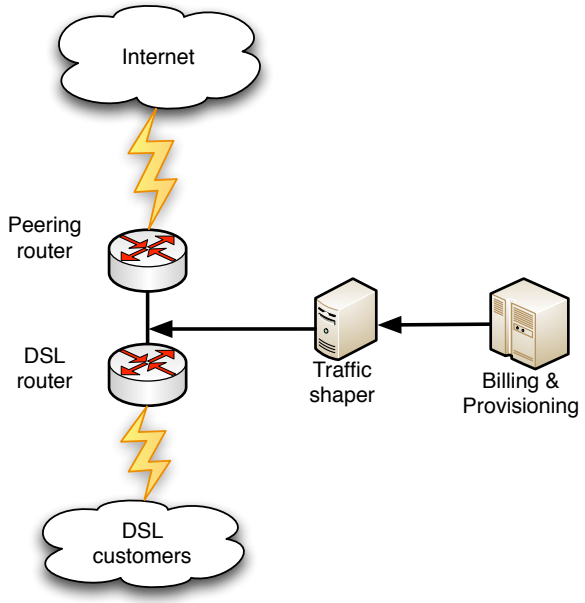


Figure 25: Case study of HAP promotion (applied by one network operator).

For example, if 20% of peers can be promoted, the operator could set $(r_1, r_2) = (0.5, 0, 4)$. An example with $N = 10$, $r_1 = 0.6$, and $r_2 = 2/3$ is shown in Figure 24. Here, the first metric selects six peers instead of four, while the second metric limits the selection to the desired four peers.

Measurement History

The HAP selection algorithm must also consider that the behavior of peers and, therefore, the measured values, might fluctuate over time, either due to random and diurnal effects, or because of changes in behavior of single users. The usage of historical data can alleviate the effect of such fluctuations, but older values should receive lower weights to allow for long-lasting changes in user behavior and in order to keep the entrance barrier low for new peers.

We choose the *modified exponential moving average* [122] to aggregate historical data with exponentially diminishing weights. Given a certain HAP selection metric R for the current measurement interval t, this results in the aggregated rating R' with:

$$R'(t) = \begin{cases} \alpha \cdot R(t) + (1-\alpha) \cdot R'(t-1) & \text{for} \quad t > 0 \\ 0 & \text{for} \quad t = 0. \end{cases}$$

A useful property of this metric is that the network operator has to store only a single historical value per peer, resulting in reduced storage and computational complexity.

The value of α depends on the interval duration τ and the user behavior fluctuations. For our evaluation, we set $\alpha = 0.5$, so that newly measured value R is taken into consideration with the same weight as the last determined $R'(t-1)$ value for a peer.

5.5.3 *Changing User Access Profiles*

The incentive-based traffic management technique described in this chapter takes advantage of the capabilities provided by Next Generation Networking (NGN) compati-

ble equipment, which, among other features, allows on-the-fly automated updates of the customer access profiles [70]. In our case this is the increase of the totally accessible upload and download bandwidth of certain users. While there are different operator- and vendor-specific solutions possible, we present one realistic example where our approach was successfully implemented.

The resulting network topology is presented in Figure 25. In this case study, the customer access bandwidth is not limited by the DSLAM but is throttled by the use of a customized Linux-based traffic shaper. That means that the bandwidth provided to single users by the DSLAM is one or two times higher than the shaped bandwidth. Such an architecture, that is otherwise used to provide triple-play or similar services that require QoS enforcement for different types of traffic, can be easily reused for our approach. Thereby, the billing and provisioning subsystem stores the current bandwidth assigned to single peers and traffic shaper enforces such limits. The applied limits can be reconfigured dynamically to promote certain customers to HAPs. This case study demonstrates the feasibility of the proposed approach, where the reconfiguration of clients takes place at least once per day.

Local vs. Global Bandwidth Increase

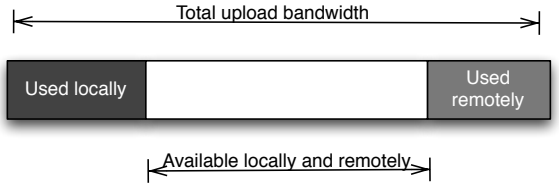
So far, we considered increasing the users' bandwidth *globally*, which means that the network operator cannot control the way in which the users use the additional bandwidth. Indeed, peers promoted to HAPs might upload a lot of traffic to remote peers and, therefore, even increase the outbound inter-domain traffic. Even if such peers might be demoted after the next HAP selection round, the new set of peers might exhibit the same behavior. The reasons for such network-unfriendly behavior might be diverse, including the lack of locality awareness in the overlay, deficit of available bandwidth (or desired content) in other network domains. Because of this, promoting the right peers by using the proposed metrics cannot mitigate such an undesired increase of the outbound traffic. To avoid such outbound traffic increase we propose that the network operator increases the bandwidth only *locally*, that is, inside of its network domain. This is an alternative to the *global* bandwidth increase that we consider as the default option. This decision may have a significant impact on the overall performance of the incentive-based traffic management system. The expectation is that a global increase is more beneficial for the whole overlay, while a local increase is more efficient from the network operator's perspective if outbound traffic is costly for the given operator. As discussed in Section 2.3.1 some operators might pay only for inbound traffic, or even be payed by other operators for the outbound traffic.

Figure 26 demonstrates the differences between the both types of bandwidth increase. With global bandwidth increase the unused bandwidth can be utilized for both local and remote transfers, while the local bandwidth increase makes it available only for local peers. Therefore, the network operator that is concerned about possible increase of outbound traffic can safely choose the second option. This typically applies to tier-3 operators (see Section 2.3.1), while tier-2 operators might prefer the first option.

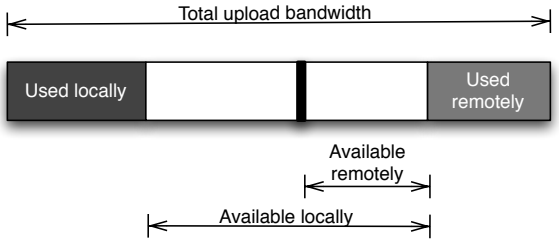
An interesting side effect of this approach is that it benefits locality-aware peers, similar to the classical traffic shaping because it makes local peers more attractive. Thus, the peers are not only incited to behave locality-aware to be promoted but they also benefit from other promoted peers.

5.5.4 *Bandwidth Dimensioning*

So far, we assumed that network operator is going to promote a subset of peers to become HAPs. The intuition behind is that only certain peers should be rewarded and that, furthermore, some peers are much more active than others (as measured for



(a) Global bandwidth increase.



(b) Local bandwidth increase.

Figure 26: Local vs. global bandwidth increase.

example by Cho et al. [31]). After we presented how the list of HAP candidates can be ranked by various metrics, the question remains: *How many of the candidates should be promoted to HAPs?*

The problem can be formulated as: Given the set of candidate peers N, choose a subset of best peers N' according to a metric R such that $\forall n' \in N'$ there is no $n \in N$ with $R(n) > R(n')$.[4] This selection is subject to the following constraints: (1) maximum possible bandwidth increase per peer u'_{max} and (2) maximum possible total bandwidth increase U' (to avoid congestion of inter-domain links). There are also similar constraints with regard to the download bandwidth, but, since the download bandwidth increase is only used to signal the promotion status to peers, the required amount of bandwidth is of lower priority.

There are at least two different ways to determine the desired subset of peers to be promoted to HAPs:

1. The network operator decides on the additional upload and download bandwidth U and D that should be allocated to all HAPs and the per peer upload increase u' and d'. Then the total number of supported HAPs is:

$$n = \min(U/u', D/d')$$

2. Alternatively, the network operator defines a rating threshold R_t and promotes all peers with higher ratings.

We opt for the first solution, since it allows the network operator to limit the potential traffic increase in its network, though, a combination of both approaches can be used that promotes up to N best peers whose ratings exceed the threshold R_t. Our choice is based on considerations of financial feasibility as every extra HAP not only leads to potential savings (due to increased locality of p2p traffic) but also brings some operational costs because of additional load on network routers and DSLAMs. Therefore, it is necessary to limit the total bandwidth increase.

Proceeding with the first solution, we focus on the upload volume as the most scarce resource from the overlay's perspective. We further define the following parameters of HAP promotion based on the total number of HAP candidates being N and u_0 being the upload bandwidth of not-promoted peer:

4 This means that R can also be a partial order on N.

PROMOTION VOLUME $U = u' \cdot n$ is the total bandwidth that must be invested by the network operator additionally to the original upload bandwidth of its customers $u_0 \cdot N$.

PROMOTION RATIO $\lambda = n/N$ is the relative number of promoted peers.

PROMOTION FACTOR $\delta = (u_0 + u')/u_0$ is the relative increase per HAP.

We can show that only two of these three parameters can be varied independently because it applies that:

$$U = u' \cdot n = u' \cdot (\lambda \cdot N) = (u' + u_0 - u_0) \cdot \frac{u_o}{u_o} \cdot \lambda \cdot N = (\delta - 1) \cdot u_o \cdot \lambda \cdot N. \quad (5.2)$$

In general, the more volume U is assigned to HAPs the higher is the potential impact of the mechanism. The appropriate value depends on the technical possibilities of the network operator (how far the upload links can be upgraded) and the capacity of intra-domain links. HAP factor δ and HAP ratio λ, in turn, should be selected carefully to assure that most relevant peers get sufficient bandwidth to change the overlay's traffic significantly.

5.6 EVALUATION METHODOLOGY

In this section, we present the performance evaluation of the proposed incentive-based traffic management approach. Our application scenario is again a commercial peer-assisted VoD system as described in Section 3.1. Consistently, our evaluation goals are:

1. to understand whether the proposed approach can result in a "win-win" situation both for the overlay and network operators,
2. to understand the impact of various parameters and optional techniques of HAP selection,
3. to assess the impact of locality awareness in the overlay and its interplay with the HAP promotion,
4. to assess the feasibility of the mechanism for a single network operator.

The operation time of HAP promotion considers long-term effects with 24 hours being the basic management interval. Therefore, we decided for long-term simulations as a suitable methodology to estimate the effectiveness of the mechanism. This distinguishes our methodology from other works, where single BitTorrent or streaming swarms are often observed over a time interval of few hours [18, 114, 92].

Because of the probabilistic nature of our simulations, random effects could bias the measured values. We address this issue by repeating each simulation run up to five times, with different random seeds controlling all random numbers generated. The result sets are then combined by computing the average values and their respective confidence intervals. The confidence level is set to 95% and the confidence intervals are included into generated plots as vertical bars for each measured value.

The rest of this section presents the evaluation setup and describes our metrics while the outcome of the performed experiments is reported in Section 5.7.

5.6.1 *Workload*

Our simulation scenario models the behavior of a peer-assisted VoD building an overlay that spans the domains of several network operators. An efficient implementation of this system in a custom discrete event-based simulator allows us to simulate a

Table 9: Evaluation setup for HAP promotion.

Parameter	*Default*	*Variations*
Peers	50,000	–
- Initial upload bandwidth	700 kbps	–
- Cache size	4 GB	–
- User groups	{heavy,casual}x{altruistic,selfish}	–
- Seeding behavior	50%: 0 minutes, 50%: 60 minutes	–
- Inter-request time	20%: 8 hours, 80%: 48 hours	–
- Diurnal pattern	idle: 12 hours, busy: 12 hours	–
- Peer selection policy	locality-aware	oblivious, mixed
Videos	2,500	–
- Size	800 MB	–
- Rate	950 kbps	–
- Popularity distribution	Zipf ($\alpha = 0.85$)	–
Network Operators	10 customer and 1 server domain	–
- Topology	Star	–
- Size distribution	see Table 10	–
- Operators employing promotion	all	none, only A, only G
Traffic Management		
- Managing interval	24 hours	6, 12, 18, 30, 36 hours
- HAP selection metric	contribution	network-friendliness, seeding ratio, random
- Promotion factor	5	2, 3, 4, 6, 7
- Promotion volume	100%	0 – 300%
- Promotion ratio	25%	100, 50, 33.3, 20, 16.7%
- Bandwidth increase	global	local, none

scenario of four weeks. This is a sufficient duration to understand whether the selection of HAP candidates and modification of their access profiles result in significant benefits.

A generalized peer-assisted application is modeled as follows: The content is initially available only on a content server. Each peer requests videos (according to a popularity distribution) and asks the indexing server for a list of video sources. The returned list contains all peers holding a copy of the video and the content server as a backup source. Possible video sources are (besides the content server) peers that already consumed the video before, did not remove it from the local cache, are online, and have free upload resources. The amount of required bandwidth for a single download is dictated by the video bitrate. The downloader always tries to download content from other peers. That is, the content server contribute only the missing upload bandwidth (calculated as the difference between the video bitrate and the bandwidth provided by other peers). This way, the overlay provider offloads it servers as much as possible and reduces its costs, while the users receive the desired QoS.

Our incentive-based traffic management approach was implemented in the simulator at a including all mechanisms described in Section 5.5. Additionally to the HAP promotion, we implemented locality awareness as an optional overlay-side traffic management mechanism. Therefore, peers (either all or a subset of them) can behave either

network-oblivious or *locality-aware* when choosing their communication partners. Since servers are considered as most costly from the overlay's perspective, with both peer selection policies, remote peers are always preferred to content servers when choosing download sources.

Table 9 shows the default setup used for simulations. The parameters are split in four groups covering different aspects of the scenario: peer properties, video properties, network operators, and traffic management parametrization. For each parameter we specify a default value, followed by the values varied in single experiments. The parameter variations are also mentioned explicitly in each experiment's description.

Peers

The overlay consists of 50,000 peers with an initial upload bandwidth set to 700 kbps. The latter value is based on a static of Ookla Net Index revealing that the median upload bandwidth of German users is 697 kbps [115]. According to the same source, the median download bandwidth is significantly higher with 6095 kbps. This is more than eight times the upload bandwidth, which is a typical situation in asymmetrical broadband networks, such as DSL. Since this value is much higher than the video bitrate, we do not consider it as a bottleneck and assume that all peers have sufficient download speed to stream videos. We do not limit the upload bandwidth of the content servers, since we are interested in their load under various traffic management configurations. Furthermore, the local caches of peers have the size of four GBs, which is sufficient to cache five videos, and apply the FIFO cache replacement policy.

We model the user behavior based on various measurement studies revealing heterogeneous usage patterns [29, 68, 67, 31]. For this purpose, we divide the users into four groups based on two independent properties: seeding behavior and inter-request time. While the inter-request time specifies how often a user consumes content, the seeding behavior describes how long a peer stays online after finishing watching a video. For the inter-request time, we apply the Pareto distribution to user requests, resulting in 20% of *heavy users* generating 80% of streaming requests, while the remaining 80% of users are considered as *casual* (similar to the patterns observed by Cho [31] and Basher [14]). We model the average inter-request times for heavy users with 8 hours and for casual users with 48 hours and distribute them exponentially. With respect to the seeding time that describes the user's level of altruism, we divide the users in *selfish* users that do not seed at all, and *altruistic* users, whose seeding times are exponentially distributed with the mean of one hour. This seeding behavior corresponds to observations in many p2p systems, such as [82].

Another relevant factor is the so-called *diurnal traffic pattern*, which describes the differences of the Internet usage during the daytime. Based on the measurements in [67], we model the diurnal traffic pattern by splitting the day into two phases: a 12-hour *busy* phase and a 12-hour *idle* phase where the request-per-day rates are increased or reduced respectively.

Last but not least, the overlay peers can use different peer selection policies that can influence their network-friendliness. As discussed in Section 3.4.4 we consider two extreme cases: network-oblivious and locality-aware peer selection that can be configured for individual peers. This allows us to specify the probability of locality-aware behavior per peer, resulting in mixed scenarios (for example, 10% of peers might behave locality-aware).

Video Content

We model the video content library with with 2,500 videos where each video is 800 MB large and has an average bitrate of 950 kbps. This corresponds to the length of 115 minutes of a full-length movie and a resolution of roughly 854x480 pixels, which is quite a conservative choice for a high-definition content. Finally, we model the content

Table 10: Distribution of peers to network operators (Based on the biggest ten German network operators according to Ookla Net Metrics [115]).

Operator	Based on	Peer percentage
A	Deutsche Telekom	32%
B	Kabel Deutschland	17%
C	Alice	13%
D	Kabel BW	12%
E	Arcor	12%
F	1&1	4%
G	Vodafone D2	3%
H	o2 Germany	3%
I	NetCologne	2%
J	freenet Cityline	2%

popularity by letting users choose videos to watch according to a Zipf distribution (with Zipf parameter $\alpha = 0.85$) similar to [20, 76]. This distribution results in some videos being much more popular in the network than the others.

Network

In order to have a realistic number of network operators and an appropriate distribution of peers to operators, we use statistics of users' bandwidth in Germany [115]. After removing many tiny operators listed there, such as single companies that would not allow employees to use a streaming application anyway, we obtain ten heterogeneous operators with the distribution of peers to operators' domains as presented in Table 10.

We further model a separate domain for the overlay provider, which contains only the content servers. The domains are organized in a star topology, as shown in Figure 27, which is sufficient for our purposes, since it allows us to capture the relevant performance metrics for single operators that are interested in reducing their inbound and outbound traffic.

Traffic Management

Our scenario further specify which operators apply HAP promotion (all, none, or single operators). The managing interval specifies both the promotion duration and the measurement interval of the monitoring component. By default this interval is set to one day, while the basic selection metric is user's contribution to the overlay. The promotion factor is set to 5 which corresponds to 400% additional upload bandwidth per peer. Finally, 25% of peers are promoted in the default setup, which results in the default promotion volume of 100% according to Equation 5.2.

In summary, every 24 hours each network operator that applies HAP promotion recalculates the contribution of its users based on the utilized HAP selection metric. Then the top 25% of its users are promoted (or stay promoted if they were promoted in the last interval). The remaining users keep their initial upload bandwidth, while those being promoted in the last interval are demoted to their initial bandwidth. Finally, the increased bandwidth is not restricted to the local domain by default (*global bandwidth increase*) and alternative restriction to the local domain (*local bandwidth increase*).

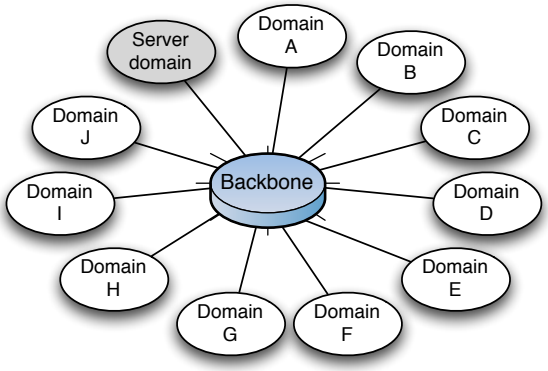


Figure 27: Network topology used in simulations (A – J are customer domains).

5.6.2 *Metrics*

We measure the following metrics to understand the impact of the mechanisms under study: (1) *traffic uploaded by servers* to capture the costs for the overlay provider (and implicitly for the overlay users), and (2) *inter-domain traffic* to capture network operators' costs. For the latter metric we also distinguish between *inbound and outbound inter-domain traffic*, since for certain operators they might have different impact on costs. We also consider the *intra-domain traffic* that does not leave the network operators' domain, as the best case with regard to the interconnection costs.

The relationship between the different types of traffic can be calculated as follows. First, we denote the total inbound and outbound traffic of non-server domains as V_{in} and V_{out}, respectively. Second, V_{in}^{d} and V_{out}^{d} denote the inbound and outbound traffic of a single network domain d. Then, we also denote the traffic amount uploaded by servers into the customer domains as V^{server}. In our setup all servers are located in a separate domain and there is no inbound traffic into the server domain because servers only upload video segments but never download them from peers. Therefore, for the total inter-domain traffic we obtain:

$$V_{in} = \sum_{d \in D} V_{in}^{d} = \sum_{d \in D} V_{out}^{d} + V^{server} = V_{out} + V^{server}$$

with D being the set of all non-server domains.

Note that we do not address the user experience directly, since in a peer-assisted scenario the servers catch up with missing resources.[5] Instead, we assume that the overlay provider transfers the distribution costs (or cost savings) to the users, by adjusting the content prices. A feasible scenario is also when the overlay provider rewards users depending on their contribution to the overlay. Therefore, a traffic management outcome that would decrease both the server traffic in the overlay and the inter-domain traffic at the network level, would benefit all three stakeholders: network operators, overlay providers, and users, and can be effectively considered as a *triple-win situation*.

5.7 EVALUATION RESULTS

In order to demonstrate the effectiveness of our approach and to understand the impact of different mechanisms and relevant parameters, we conducted a series of experiments:

5 In Chapter 4 we demonstrated how adaptive server allocation can provide this functionality.

BASIC IMPACT AND SELECTION METRICS: Our first experiment analyzes whether the HAP promotion can provide a win-win situation for the overlay and the network. We further compare the impact of HAP selection metrics presented in Section 5.5.2 and discuss their impact on the overlay performance and their preferences for certain user groups. Note that this experiment assumes a locality-aware overlay.

INCENTIVES FOR LOCALITY AWARENESS: This experiment analyzes how the overlay can be incited to behave network-friendly, since HAP promotion might benefit the overlay even if it does not behave locality-aware. For this purpose, two different sub-experiments are conducted: (1) Effect of Network-friendliness and Combined Metrics where locality-aware peers are preferred over network-oblivious peers and (2) Effect of Local Bandwidth Increase that prevents the bandwidth of promoted local peers to be utilized by remote peers. The main question of this experiment is whether locality awareness becomes a dominant strategy for the overlay, if operators deploy HAP promotion.

EARLY ADOPTER: In this last experiment we aim to understand the impact of HAP promotion on an *early adopter*. An early adopter is the first network operator that decides to apply HAP promotion, without other operators using it. Thereby, we analyze the situation of small and large operators and highlight the differences. We further compare the effect of local and global bandwidth increase for such operators and discuss the impact on the overlay performance.

Additional experiments covering the proper combinations of HAP ratio and factor, as well as the management interval are presented in Appendix A.

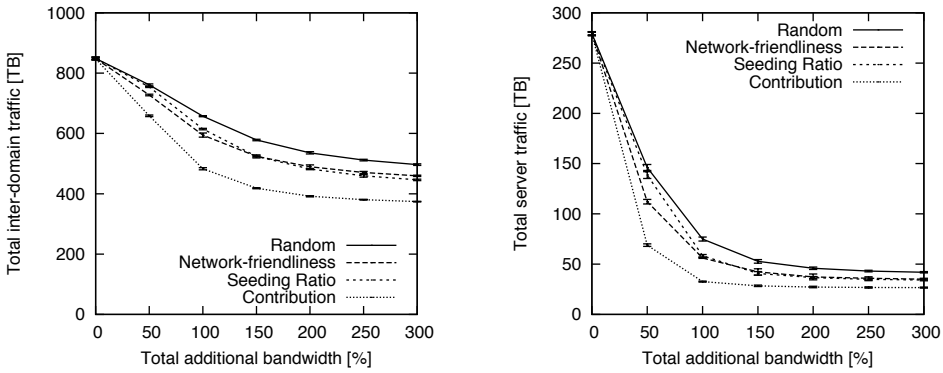
5.7.1 *Basic Impact and Selection Metrics*

The goal of this experiment is to understand whether HAP promotion can achieve a significant inter-domain traffic reduction and decrease server traffic at the same time. Furthermore, we aim to understand which is the preferred HAP selection metric from the operators perspective.

The experiment uses the setup presented in Table 9 with the variations presented in the following. At first, we assume that all peers behave locality-aware, an assumption that is dropped in the subsequent experiments. All network operators apply HAP promotion with a variable amount of additional upload bandwidth. For this purpose, we varied the amount of additional upload bandwidth from 0% (no HAP promotion) to 300% (3 times higher overall bandwidth than without HAP promotion). Each network operator promotes 25% of its peers in the interval of 24 hours and the additional upload bandwidth for HAPs is not restricted to the local domain (global bandwidth increase). Note that the variable HAP volume of 0 – 300% and a fixed HAP ratio of 25% result in a variable HAP factor (bandwidth increase per HAP) in the range $\{1, 2, 3, 4, 5, 6, 7\}$, though the feasibility of values above 3 might be restricted for some network operators (as shown in Equation 5.2).

The main algorithm under study is the HAP selection metric. We consider all four *basic* selection metrics presented in Section 5.5.2: *random*, *network-friendliness*, *seeding ratio*, and *contribution*. Since each of the metrics focuses on different aspects of peer behavior, we expect different types of users being preferred by them, which in turn should affect the performance metrics.

Figure 28 shows the major outcome of this experiment. We observe that even with the *random* HAP selection metric the inter-domain traffic can be significantly reduced from 850 to 500 Terrabyte (TB) (see Figure 28a). At the same time the server traffic can be reduced from 280 to 45 TB (a reduction of 85%), which results in the intended *win-win* situation.



(a) Inter-domain traffic. (b) Server traffic.

Figure 28: Impact of single HAP selection metrics (25% of peers promoted).

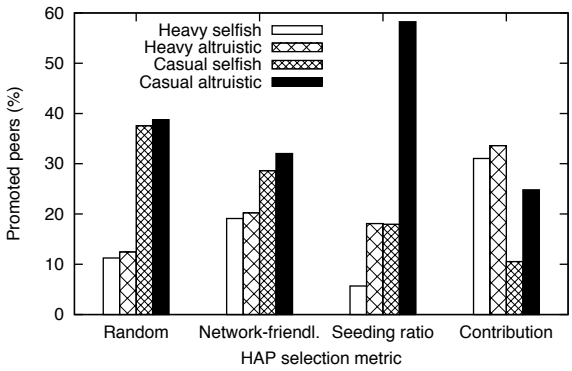


Figure 29: Type of users promoted by single HAP selection metrics.

Besides this naïve metric all three other metrics perform even better, resulting in higher benefits both for the overlay and network. *Contribution* turns out to be the best HAP selection metric in all setups, reducing the inter-domain traffic by up to 55% and the server traffic by up to 90%. The other two metrics, *Network-friendliness and Seeding ratio*, range closer to the random metric.

An additional interesting result is that even if the operators invest only 100% of additional upload bandwidth, the savings are still significant, being 43% for the inter-domain traffic and 88% for the server traffic with the best metric. This increase of the total upload bandwidth by 100% is considered as our *working point* for the next experiments.

But why was the contribution metric able to outperform the other metrics? To answer this we presents the percentage of promoted peers within single behavioral groups (for a single simulation run) in Figure 29. Here, the random metric shows no preference per user type (the presence of each user type is the same as in the total population, while heavy selfish users constitute 10% of the overlay population). On the other hand, the network-friendliness metric slightly prefers heavy users by promoting up to 20% of them (both for selfish and altruistic subgroups) but it does not reward altruistic behavior. Furthermore, the seeding ratio metric prefers mostly altruistic users (and herein casual users with almost 58%), and only then heavy users. Finally, the contribution metric prefers heavy users to casual but also prefers altruistic behavior in the second place (casual altruistic users constitute 25% of the promoted users while casual selfish constitute only 11%, though the size of both groups is the same).

The obtained results confirm our expectation of the mutual benefit provided by the HAP promotion. Also we observe that the focus on heavy users (and then altruistic ones) as done by the contribution metric achieves best results for the overlay and the network. Furthermore, it turned out that the total bandwidth increase of 100% results in significant improvements with regard to our performance metrics.

5.7.2 *Incentives for Locality Awareness*

So far, we assumed that 100% of the peers behave locality-aware and we saw that, in this case, HAP promotion benefits both parties. In this experiment, we assess what happens if we drop this assumption. The major question is whether the HAP promotion mechanism can incite the overlay to behave locality-aware?

Our experimental setup is based on the previous experiment, with all operators applying HAP promotion at the working point, that is, when the total upload bandwidth is increased by 100%. In order to assess the impact of locality awareness on the performance metrics, we vary the percentage of locality-aware peers between 0 and 100%.

Two relevant mechanisms are used to incite locality-aware behavior: (1) promotion of locality-aware peers via the *network-friendliness metric* (see Section 5.5.2) and (2) local bandwidth increase for promoted peers (see Figure 5.5.3). For the first mechanism we vary the applied HAP selection metric, while using the global bandwidth increase. Here, the random HAP selection and contribution are valid comparisons, as in the previous experiment they have shown the best and worst performance for both parties. For the second mechanism we compare the effect of global and local bandwidth increase.

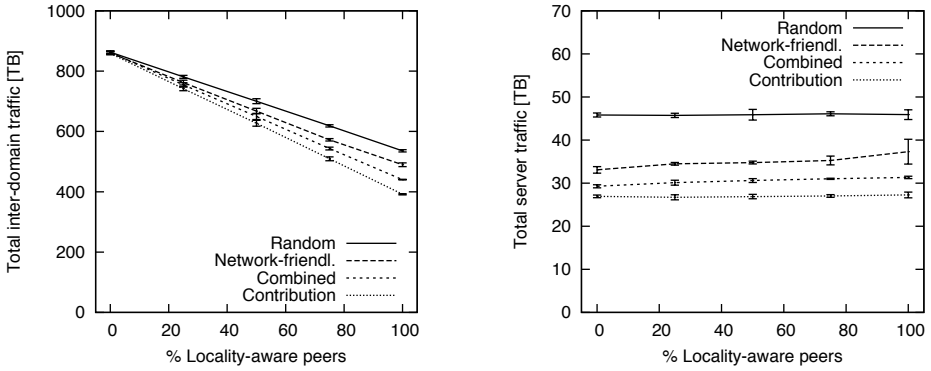
The HAP selection based on network-friendliness is expected to incite users by promoting locality-aware users more often than other non-local metrics, while the local bandwidth increase is expected to incite the overlay provider by higher reduction in server traffic with increasing locality awareness.

Effect of Network-friendliness and Combined Metrics

The effect of the network-friendliness metric and its combination with the contribution metric is demonstrated in Figure 30. It shows that the *combination of contribution and network-friendliness metrics offers a trade-off between pure contribution and pure network-friendliness metrics*, both for the overlay and network. We can see that in case of a network-oblivious overlay, when the percentage of locality-aware peers is equal to zero, all metrics perform the same (see Figure 30a), while with the increasing locality awareness degree the inter-domain traffic decreases. This confirms our expectation that locality awareness in the overlay amplifies the effect of HAP promotion. Furthermore, the contribution metric is again the best one and random selection performs worst. The pure network-friendliness metric ranges in the middle, while the combination of contribution and network-friendliness metrics comes closer to the pure network-friendliness metric.

The effect is very much the same for the server traffic, where the combination of contribution and network-friendliness metrics is slightly worse than the pure contribution metric, but better than the other metrics (see Figure 30b). Furthermore, we see that the degree of locality awareness has no direct impact on the server traffic, since all lines are almost horizontal.

In order to understand the impact on users, we plot the percentage of promoted *locality-aware* peers vs. their total percentage in the overlay (see Figure 31). We observe that with both selection metrics based on the network-friendliness the probability to be promoted is higher for *locality-aware users* because their percentage in the promoted peers is higher than their percentage in the overlay population.



(a) Inter-domain traffic.

(b) Server traffic.

Figure 30: Combined vs. single HAP selection metrics (varied degree of locality awareness in the overlay).

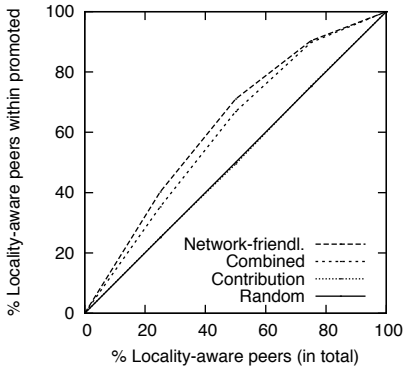


Figure 31: Percentage of locality-aware peers within promoted peers (random and contribution metrics overlap).

Additionally, Table 11 depicts the probability of being promoted for both groups of users (locality-aware and oblivious). The probability $\mathcal{P}$ is computed for each group G as

$$\mathcal{P}_G = \frac{|G \cap H|}{|G|},$$

where H is the set of peers promoted to HAPs. According to our data, if 50% of users are locality-aware, 38% of them are promoted with the network-friendliness metric, but only 13% of network-oblivious users. The effect is also very similar when the combined metric is used (33% vs. 17%). This provides a *clear incentive for users to behave locality-aware* because the promotion yields a better standing in the overlay and is associated with a simultaneous increase in download bandwidth.

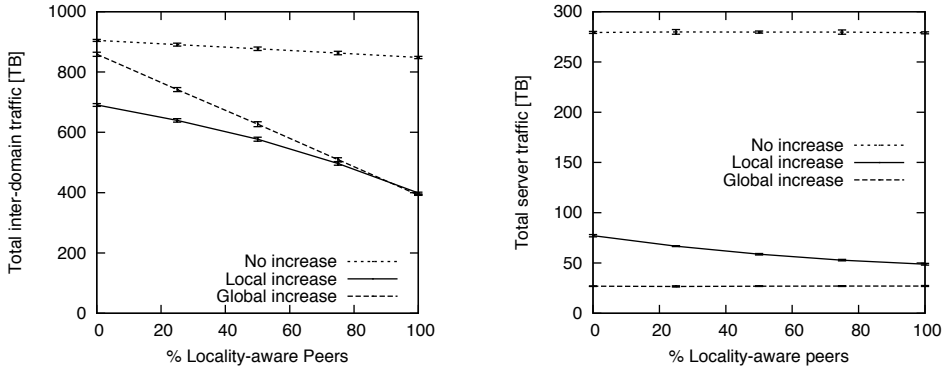
To sum up this experiment, network operators should include network-friendliness in the HAP selection algorithm to incite users. Here, especially the combination with the contribution metric provides best results, since it prefers locality-aware peers similar to the pure network-friendliness metric, while reducing inter-domain traffic and server traffic similar to the pure contribution metric.

Effect of Local Bandwidth Increase

In this sub-experiment we again vary the percentage of locality-aware users. This time we fix the HAP selection metric to the contribution but vary the type of bandwidth increase between *local* and *global*. We also include the case when no bandwidth is

Table 11: Promotion probability for locality-*aware* and network-*oblivious* peers depending on their ratio (within the population) and applied HAP selection metric (The *combined* metric denotes the combination of the contribution and network-friendliness metrics).

Locality-aware peers (%)	HAP Selection Policy and peer group							
	Random		Network-friendliness		Combined		Contribution	
	aware	obliv.	aware	obliv.	aware	obliv.	aware	obliv.
0	–	0.25	–	0.25	–	0.25	–	0.25
25	0.25	0.25	0.40	0.20	0.35	0.22	0.25	0.25
50	0.25	0.25	0.38	0.13	0.33	0.17	0.25	0.25
75	0.25	0.25	0.30	0.10	0.30	0.10	0.25	0.25
100	0.25	–	0.25	–	0.25	–	0.25	–



(a) Inter-domain traffic.

(b) Server traffic.

Figure 32: Local vs. global bandwidth increase (applied by all network operators).

increased to demonstrate the impact of plain locality awareness without HAP promotion.

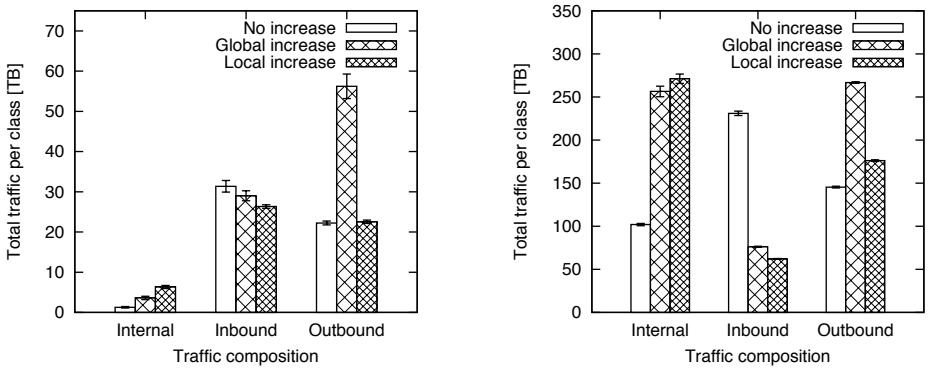
The results of this experiment are presented in Figure 32. Without HAP promotion, we observe only a minor benefit of locality-aware peers for network operators and no impact on the server traffic. On the other hand, the global bandwidth increase benefits the operators, but the overlay provider again observes no difference with regard to the server traffic. Finally, the local increase reduces the inter-domain traffic more effectively, especially if most peers are not locality-aware.

For the overlay, the global bandwidth increase is always better but with the local bandwidth increase a higher locality awareness in the overlay also implies additional server traffic reduction. Finally, the cooperation of overlay and network achieves the best result for the network operator (when 100% of users behave locality-aware and the network operators increase the bandwidth globally).

We conclude this experiment with the observation that *local bandwidth increase does indeed incite the overlay provider to promote locality awareness in the overlay*. In this case, the server traffic can be further reduced from 150 TB without locality awareness to only 50 TB with fully locality-aware overlay.

5.7.3 *Early Adopter*

In the previous experiments either *all or none* of the network operators containing overlay users were applying HAP promotion. While, in this situation, the network as a whole was able to benefit from our approach, the question arises whether it is also



(a) Small network operator (G).

(b) Large network operator (A).

Figure 33: Performance of an early adopter (either large or small operator).

beneficial for an early adopter, that is, the first operator applying HAP promotion. Only if this step is beneficial for a network operator, we can expect the mechanism to be deployed.

To assess the situation of such an early adopter operator, we let only a single operator, either a small operator (G) or a large operator (A), apply HAP promotion within its own network domain. Additionally to the server traffic, we capture the amount of inbound and outbound traffic for the operator applying HAP promotion to assess the outcome.

The basic configuration of the HAP promotion stays the same as in the previous experiment. Besides the choice which operator applies HAP promotion the type of bandwidth increase, either local or global is the only variable parameter.

Our choice of the operators to be an early adopter is dictated by the expectation that a large operator can more easily benefit from the mechanism because of the potentially higher impact on the overlay as a whole and a larger number of local peers (and also a higher availability of content locally). Another expectation is that local bandwidth increase is able to prevent an increase in the outbound traffic.

Impact on Network Operators

Figure 33 shows the outcome of this experiment for both of the potential early adopters. Our first observation is that the absolute changes are much stronger for the large operator (as expected). Since the trends for all kinds of traffic are the same, in the following, we focus on the large operator A.

Comparing the effect of introducing the global bandwidth increase with the no increase case that effectively means no HAP promotion at all, we observe for the big network operator (see Figure 33b):

- Intra-domain traffic increase by 151.6% ,
- Inbound traffic decrease by 67,1%,
- Outbound traffic increase by 83.5% that almost cancels the inbound savings.

The situation is very much the same for the small operator G with the exception that the improvements in intra-domain and inbound traffic are much smaller compared to the outbound increase (see Figure 33a). This happens due to the higher utilization of additional *local* capacities by *remote* peers.

We further observe that the local bandwidth increase is able to alleviate the situation of the early adopter compared to the global bandwidth increase where the total inter-domain traffic is not reduced but only the inbound traffic. The outbound traffic is

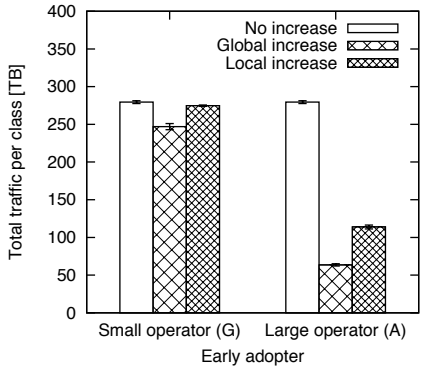


Figure 34: Impact of an early adopter (operator A or G) on server traffic.

now at a similar level as before, while the inbound traffic savings are even slightly higher compared to the situation without HAP promotion. With the local bandwidth increase the large operator is now able to reduce its total inter-domain traffic from the original $231 + 145 = 376$ to $62 + 176 = 238$ TB , compared to $76 + 267 = 343$ TB with the global bandwidth increase (see Figure 33b). The situation is again very similar for the small operator where the local bandwidth increase reduces its inter-domain traffic from $31 + 22 = 53$ to $26 + 22 = 48$ TB (see Figure 33a).

Impact on the Overlay Provider

Consecutively, we consider the impact of case of HAP promotion applied by the early adopter on the overlay provider. Figure 34 demonstrates that the overlay benefits in both cases and with each early adopter, though from the large early adopter more than from the small one. This is consistent with our expectations, since the large operator hosts ten times more peers than the small one, and promoting 25% of them to HAPs results in a larger performance boost with regard to the bandwidth available in the overlay. We further observe that the local bandwidth increase results in lower savings than the global one because in the first case the overlay peers located outside of the early adopter's domain cannot use the additional bandwidth of promoted peers.

In summary, the evaluation of the early adopter case shows that a network operator must be careful how to increase the bandwidth of promoted peers. When other network operators do not implement HAP promotion, an early adopter may even suffer from the deployment, if outbound traffic is costly. The reason is that the overlay might use additional available resources introduced by the early adopter in the undesired manner by uploading to remote peers. However, we showed that this drawback can be mitigated by increasing the upload bandwidth only for the local transfers. Then the network operator can fully profit from the benefits of incentive-based traffic management, regardless of whether other network operators deploy it or not.

5.8 SUMMARY

This chapter presented a novel mechanism for overlay traffic management. A network operators applies the mechanism, called Highly Active Peer (HAP) promotion, to reduce its interconnection costs. The mechanism provides incentives to the overlay provider and users to behave network-friendly for the mutual benefit of the overlay (provider and users) and network operators. For this purpose, the network operator promotes selected peers by increasing their upload bandwidth. The increased capacity of the promoted peers improves the overlay performance and increases the attractiveness of local peers. The overlay has the freedom to behave locality-aware that would

result in reduced interconnection costs for the network operators. The network operator rewards locality awareness by preferentially promoting network-friendly peers.

The considered application scenario was a peer-assisted VoD system, where an overlay provider tries to minimize the load on its servers while providing high streaming experience to the users. Our evaluation has demonstrated the effectiveness of the approach, showing a significant reduction in inter-domain traffic and an improved overlay performance. We further saw that network operators should select peers (to be promoted) based on their contribution to the overlay and network-friendliness. An additional measure is the local bandwidth increase, which can prevent an abuse of additional bandwidth and provides an additional incentive for locality-aware behavior. Finally, we demonstrated that the mechanism can be effective, even if only a single network operator adopts it.

To conclude, our approach is able to fulfill the requirements stated in Section 5.3:

1. *Reduction of inter-domain traffic* is achieved in most scenarios as shown by the evaluation. This always applies to the inbound traffic. We also presented additional mechanisms that take the specific requirements of some network operators into account by avoiding a simultaneous increase in the outbound traffic.

2. *Network operators apply* the mechanism and can effectively *control* it to provide benefits only to compliant overlays and users.

3. Overlay providers and users can benefit from this mechanism, which *provides clear incentives* for cooperation with network operators.

4. We focused on heavy hitter applications, such as peer-assisted streaming applications, which promises a *high impact* due to the increasing significance of video distribution.

5. The approach is *protocol-independent*, since it does not rely on specific protocols or mechanisms in the overlay.

6. The approach is *not content-aware* because it requires the operators to monitor only the total traffic volume per user and destination networks of user traffic, instead of the content itself.

7. The approach is *compliant with the network neutrality* principle (according to the considered definition) because it does not hurt the performance of any application.

Next chapter demonstrates how a peer-assisted VoD system can efficiently use set-top boxes in terms of their online times. The chapter also considers the interplay of peers' online times and the amount of the inter-domain traffic generated by online peers.

6 ENERGY-AWARE UTILIZATION OF SET-TOP BOXES

In this chapter, we consider the delivery costs of a peer-assisted VoD system based on Set-top Boxes (STBs) from the users' perspective. The existing approaches for STB-based architectures often assume that users leave the devices switched on most of the time. While this behavior enables the overlay provider to offload its servers, the baseline power of STBs is wasted. Therefore, we aim to manage the online times of STBs more efficiently. First, we present the common policies used in the related work and identify their limitations. After a formulation of the desired properties, we devise two adaptive standby policies. an optimal policy, and a heuristic. Based on realistic data, we assess the trade-off between the performance and energy cost of such systems, thus, taking the interests of overlay providers and users into account. Finally, we present and evaluate a network-friendly extension to the proposed policies.

6.1 MOTIVATION

An overlay provider's main goal in utilizing an STB-based delivery network is to save bandwidth and to reduce hosting costs of its servers. Additionally, this approach reduces the cooling costs of the servers that make up a large part of hosting costs [148, 91].

The reduction of server costs (without a performance degradation) requires that users contribute sufficient amount of resources. These resources include the upload bandwidth, CPU, memory, disk storage, and last but not least their *online time*. As Internet access is typically paid in a flat-rate manner, the additional upload of video segments might not result in additional costs for the users. The requirements of CPU, memory, and disk storage are also comparably low, since the uploaders don't need to render or re-encode the videos. Instead, the only requirement is to forward video segments to other receivers.

However, excessive online time may result in energy wastage, especially, if the STBs[1] stay always online, though being idle to a large extent. While some of the recent proposals for STB-based systems rely heavily on STBs being online most of the time [60, 148, 24], our goal is to reduce the STB's energy consumption by eliminating idle times. This must be done carefully to avoid severe performance degradations.

While the baseline power consumption of STBs is assumed to be low compared to desktop computers (for example 12-55 watt for typical STBs vs. about 75 watt for desktop computers [102]), it still results in annual rates comparable to electric ovens and refrigerators, exceeding the annual consumption of desktop and laptop computers [63]. The more powerful the STBs become, the higher power consumption can be expected, as in the case of game consoles consuming easily 150 watt in the idle state [21]. Therefore, an STB-based content delivery architecture demands advanced mechanisms that offload content servers but don't require STBs to be always-on.

6.2 RELATED WORK

The idea of using STBs as a platform for peer-assisted content distribution has been proposed by several researchers. Several works deal with video streaming by utilizing STBs or home gateways, focusing on the specifics of live streaming [60, 24], or video-on-demand streaming [74, 27, 142, 148].

1 Within this chapter all peers are assumed to be STBs and, therefore, we use both terms interchangeably.

Most works on STB-based systems assume that a network provider controls a set of gateways within the own network domain with gateways being always-on and having plenty of bandwidth resources [142, 60]. This makes the content cached on STBs highly available, allows proactive content replication, and (in the case of live streaming) facilitates tree-based application level multicast. However, alternative deployments require energy-aware solutions, where STBs cooperate to deliver content on behalf of various overlay providers across domain boundaries. In real networks, the bandwidth is not plentiful and energy costs for users are important, which makes the always-on behavior undesirable and, therefore, reduces content availability.

Laoutaris et al. proposed to replace traditional data centers with provider-managed distributed STB networks instead of PC-based p2p systems [90]. The authors argue, that such *Nano Data Centers* (home gateways or STBs), located at the edge of the Internet, cooperating in a peer-to-peer manner but controlled by a single authority, can expose higher security, Quality of Service (QoS), and coordination, which make them useful for VoD streaming, online gaming, and interactive TV. Controlled by the network operator that manages these boxes, the system should be able to provide similar service guarantees as server-based systems at reduced costs, since service availability requirements for traditional data centers result in considerable redundancy with respect to network access, storage capacity, and power supply. This work provides a strong motivation to hybrid platforms, combining the benefits of p2p and traditional server-based architectures in the STB scenario. However, the proposed deployment focuses on the case where the network operator controls the entire delivery network, which eliminates the need for inter-domain traffic. In our work we extend this idea to the case where the overlay provider manages an STB-based system spanning domains of multiple network operators. Furthermore, the authors propose that the provider incentivizes the users to leave their boxes on, an approach that is contrary to recent *Green IT* initiatives [108].

Further, Janardhan and Schulzrinne propose a *localized* peer-assisted streaming system based on IP-enabled STBs, such as Tivo DVR [74]. STBs are centrally controlled through a common aggregation router within each geographical area, which provides a sort of locality awareness. Movies are split into chunks and distributed via a BitTorrent like protocol. While considering the network locality issue, this work does not address the energy consumption of STBs explicitly.

A popular research aspect of STB-based systems is the proactive content placement on idle STBs to satisfy current and future user demand, based on the assumption that STBs are online 80-90% of the time [26, 27, 148]. The resulting systems "free-ride" on already dedicated baseline resources in order to reduce content servers' load [27] or their energy consumption [148]. The latter work by Valancius et al. assumes that STBs might optionally support a standby state but consider only a simple selfish policy (where users switch off their STBs after finishing their downloads), instead of adaptive policies proposed by us. The always-on assumption might hold for home gateways (such as DSL modems with hard drives), but not for IP-enabled Digital Video Recorders (DVRs), game consoles, and other entertainment devices where users are tempted to switch them off when they don't use them actively. The latter *selfish* behavior allows users to save the baseline power of STBs but also reduces their availability and the savings of servers' energy.

Different approaches were made to reduce the energy consumption of Internet-wide content delivery in order to make it "greener" [126, 44, 125]. They include the reduction of server's idle power [103] and baseline power consumption by the so-called proportional computing [12]. For example, Lee et al. have shown that the energy consumption of STB-based content delivery systems might be even higher than that of traditional server-based systems [91]. They propose a combination of energy-proportional computing and content-centric networking with routers caching the content close to the users. However, this requires the introduction of content routers that

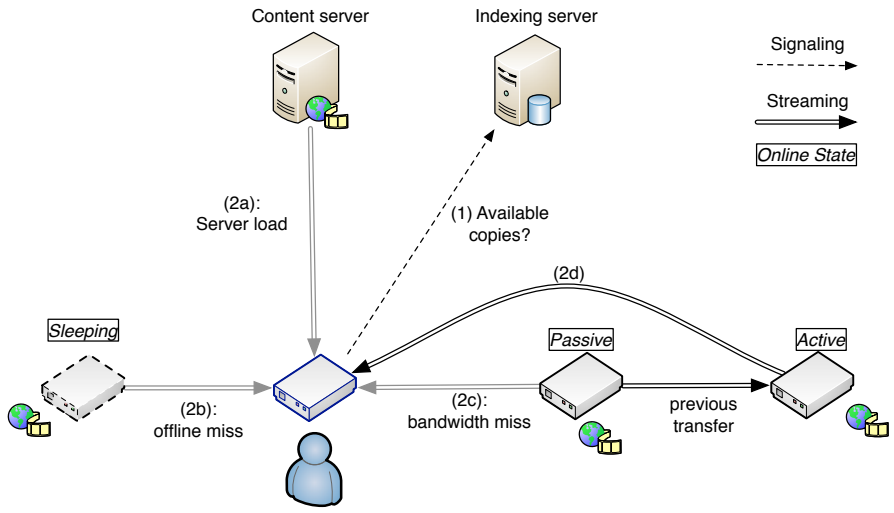


Figure 35: System architecture. A video request results in four alternative source types, where only (a) and (d) are successful.

must be aware of the stored content, which leads to a more complex and content-aware Internet infrastructure.

This analysis strengthens our belief that STBs must become more competitive by dropping the assumption that STBs must be online most of the time. We show the benefits and challenges of adaptive standby policies that can be used to achieve this goal. In an independent work Jourjon, Rakotoarivelo, and Ott investigate a distributed algorithm to reduce the overall energy consumption of an STB-based system [77], which, however, does not present any performance results of proposed algorithms.

6.3 SYSTEM MODEL

Though an STB-based architecture can be used to offer various services and applications, this thesis focuses on video-on-demand streaming. The considered hybrid content delivery network combines centralized and decentralized entities as already described in Section 3.1. However, because of the usage of STBs acting as peers there are certain differences resulting from the STB-specific features: *reliability* and *controllability* (through the overlay provider). STBs are more reliable than traditional computers because they are dedicated to only few features (such as video playback and recording, exchange of video segments, etc.) opposed to the desktop or laptop computers where multiple applications compete for resources with each other. Furthermore, the controllability enables the utilization of self-managed standby policies instead of manual online time management.

Figure 35 shows a typical streaming request in the STB-based system. In order to download a video, the requesting peer first connects to the indexing server and receives a list of STBs that already downloaded this video in the past and still have its replica in their local caches. Optionally, the reply can also include additional information about the *online state* of these peers and their load. The video can be streamed from multiple sources (STBs or content servers) by dividing the video into sub-streams that are combined into single stream at the receiver side. In this regard, the totally served rate must be at least the bitrate of the video to enable high streaming experience for the users. If the STBs caching the video cannot satisfy the rate demand (that means their available upload bandwidth is lower than the video bitrate) the requester asks a content server to provide the missing substreams.

There are several reasons why an STB might not be able to upload a previously consumed video leading to the different types of misses as discussed in the following.

CACHE MISS: The STB has already removed the video from its local cache.

OFFLINE MISS The STB has a video replica in its cache, but is offline.

BANDWIDTH MISS: The STB has a video replica in its cache and is online, but has no free upload bandwidth because the whole bandwidth is already dedicated to other peers. Thereby, the upload bandwidth might be used for the same or other videos.

Since the main goal of the system is to offload the content servers as much as possible, different mechanisms are applied to avoid the aforementioned types of misses in order to increase the percentage of data uploaded by peers. This way both the total data volume to be served by content servers and their peak load can be reduced. In Section 3.4.4 and Section 3.4.5 we already discussed peer selection policies and cache replacement policies that can be used to avoid bandwidth and cache misses, respectively. In the following we focus on the suitable measures to prevent online misses.

6.4 ONLINE TIME MANAGEMENT

The amount of offline misses can be reduced by increasing the availability of online peers. This availability can also reduce the amount of bandwidth misses, since more upload bandwidth is offered if more peers stay online. However, too excessive online time might result in too many offered video replicas and, consequently, extensive idle times.

In order to consider the availability of peers, we distinguish different *online states*, which effect STB's ability to serve other peers' requests (see Figure 35). Thus, the STB's state depends on its current operation, resulting in different power consumption levels and varying availability as summarized below:

- STB is in the *active* state while requesting and downloading a video from multiple sources in parallel. Thereby, it can be optionally forwarding the same video or upload previously cached videos to other peers. This state is unavoidable for STB's basic functionality but often exposes the highest power consumption.
- STB is in the *passive* state if it is uploading data to other peers without downloading. This state is desirable from the providers perspective, since uploads from STBs help to offload content servers. It is also tolerable (or even desirable) from user's perspective, if the provider offers a reward for data upload. Typically, the power consumed in this state is very close to that of the active state.
- STB is in the *sleeping* state if it does not provide any service (and also cannot reply to new download requests) but allow the user to boot it up within few seconds. This state exhibits the lowest power consumption and is, therefore, desirable from users perspective, even if the user is rewarded for data upload to other users. From the overlay provider's perspective this might be unfavorable, since the cached video replicas become unavailable.
- Finally, the STB can be in the *idle* state while not downloading or uploading any videos but being able to serve requests from other peers. This state can result, for example, from the always-on behavior of STBs, and make up a major part of their online times and overall energy consumption. Unfortunately, many STBs consume almost the same power in this state as in the *active* and *passive* states [148, 77]

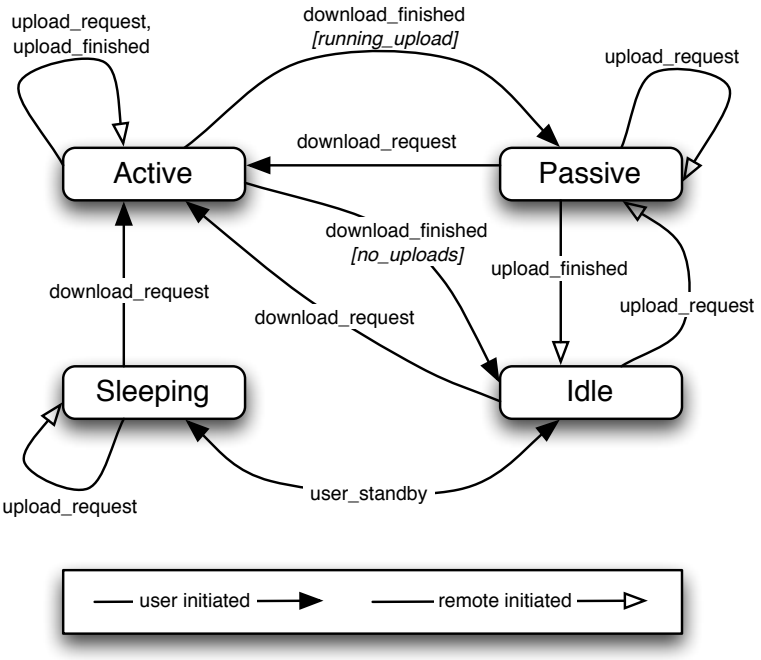


Figure 36: State transition diagram (without standby policies). *[no_uploads]* and *[uploads_running]* specify whether the STB is still serving further upload requests.

Figure 36 shows the state transition diagram in the case, where the user controls the switching into and from the *sleeping* state. The state transitions depend either on local events (download requested or finished) or remote events when other STBs request streams from this STB (upload started or finished). Furthermore, the user has to decide *manually* whether (and when) to switch the STB between the idle and sleeping states.

In the following, we consider some typical policies how the online time of STBs can be managed, either manually by the users or by the built-in functionality of STBs. Later, in Section 6.5, we devise alternative policies.

Always-on Policy

In this extreme case the user does not switch the STB off at all. As a consequence, the STB is never in the sleeping state and stays online once it joins the system. From the overlay provider's perspective this *always-on* policy is the simplest and most beneficial policy, since it assures that all copies cached by STBs are available in the overlay. Due to the permanent availability of all peers, such a policy allows for the highest possible peer contribution, and, in consequence, for the highest load reduction at content servers. However, it exposes the highest online time and, therefore, energy consumption, becoming unacceptable for the users in a long-term. In fact, such a policy shifts the energy costs from the overlay provider (by reducing the number of online servers) to the end-user.

Selfish Policy

Another extreme case, the so-called *selfish* policy, governs STBs to stay online only as long as they are consuming some resources from the system such as watching a video. As a consequence, STBs are never in the idle state. Once a user finishes the running downloads and switches off the STB, its resources become unavailable to the system.

The shutdown can be done either manually or via a built-in mechanism of the STB. Since STBs can contribute their upload resources while they are online, we expect that such a policy results in much higher server traffic compared to the always-on policy. Additionally, this policy results in a minimal idle time, in fact being zero.

Overtime Policy

In reality, the users rather tend to keep their STBs online partially, for example, sending them into sleeping state some time after they finish using it actively. The reasons are either some level of altruism, obliviousness, or the knowledge that in the passive state they contribute to the system and get a reward. However, we cannot realistically expect users to know whether their STB is currently needed and switch between idle and sleeping state based on the overlay performance. Therefore, we assume the *overtime* behavior where users keep their devices online for a fixed *seeding time* (for example one hour) after finishing their downloads.

The impact of this policy on the system performance strongly depends on the concurrency of user requests. If most users perform requests almost in parallel, moderate seeding might suffice to achieve high server traffic reduction. However, the non-adaptive nature of this behavior might also result in unnecessary idle times or high server traffic.

6.5 ADAPTIVE STANDBY POLICIES

In the previous section we already saw several *non-adaptive* standby policies for STBs: always-on, selfish, and overtime, which do not adapt to system's requirements but behave statically. In this section, we formulate the requirements for *self-managed and adaptive* standby policies and then propose two alternative policies based on specific features of controllable STBs.

In this regard, we aim to reduce the energy consumption of STBs in the system by applying an *adaptive standby policy* that governs when idle peers can switch into the sleeping state and back. Such a policy can reduce the number of offline misses while allowing peers to save energy if there is a good chance that it will not result in unnecessary server traffic. This allows reducing STBs energy consumption and, therefore, costs for end-users. Especially the idle state is undesirable, since neither the users nor the overlay provider can benefit from it.

We define a *standby policy* as a rule that governs the switching between the idle and sleeping modes of an STB. According to this definition, the selfish, always-on, and overtime policy can be considered as non-adaptive standby policies, since they don't react on the system's performance. However, all three of these policies can be easily realized in STB's firmware or executed manually.

A properly designed (adaptive) standby policy should fulfill the following requirements:

AVAILABILITY: This requirement states that the content available in the overlay should be served from peers. The less offline misses occur, the less video segments must be served by content servers that serve segments due to the bandwidth and cache misses. In the optimal case, a standby policy should avoid offline misses completely.

EFFICIENCY: This requirement demands that unused STBs are switched into the sleeping state if their services are not required. This way the the idle time of STBs and the associated energy consumption should be minimized.

We can easily see that none of the non-adaptive policies fulfills these requirements, because the selfish policy violates the *availability* requirement, the always-on policy the *efficiency* requirement, and the overtime policy might even violate both of them.

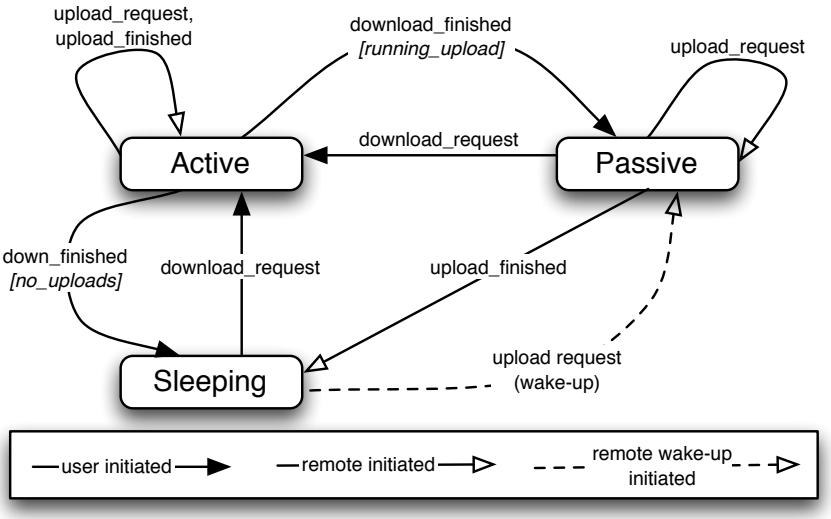


Figure 37: State transition diagram of the wake-on-demand policy.

The next section presents our standby policies to avoid STBs' lingering in the idle state and discusses the benefits and drawbacks of each policy. At first we present a policy that achieves the optimal behavior, though being rather difficult to implement in a real system. Yet, it allows us to establish the best case for comparison with alternative policies. Then we present a heuristic standby policy that tries to match online time of STBs with the demand based on the current load of the system and resources supply through online STBs. We further discuss the overhead of the proposed policies in terms of additional reports required to manage the content availability in the overlay, and present a locality-aware extension of the heuristic standby policy.

6.5.1 *Wake-on-Demand Policy*

According to the previously stated requirements, an optimal policy must eliminate idle times by sending nodes into sleep if they have finished all running uploads and downloads. At the same time an optimal policy must assure that an STB goes online if it should upload cached content. The latter condition is fulfilled when there are no online STBs that can upload the desired content.

Therefore, we devise a *wake-on-demand* policy, which switches idle STBs into the sleeping state immediately after finishing downloads and current uploads (see Figure 37). If another peer tries to download a video that cannot be provided by online STBs, the indexing server computes the missing bitrate and selects one or several sleeping peers owning a replica of the desired video. The policy then *wakes up* those peers (dashed arrow in Figure 37), which then provide the desired bandwidth and switch back into the sleeping state after finishing the upload. We expect this *wake-on-demand* policy to exhibit the optimal performance because it assures that *all required* peers are available when they are needed while eliminating idle times altogether.

Considering the selection of peers to be woken up out of the candidate list (that is maintained by the indexing server based on the information provided by the peers), we apply the same *peer selection* policy as for the non-sleeping candidates (cf. Section 3.4.4). The information about the available online and sleeping candidates is managed based on peer reports (see Section 6.5.3 for more details). This allows the indexing server to determine which of the sleeping STBs possess a replica of the de-

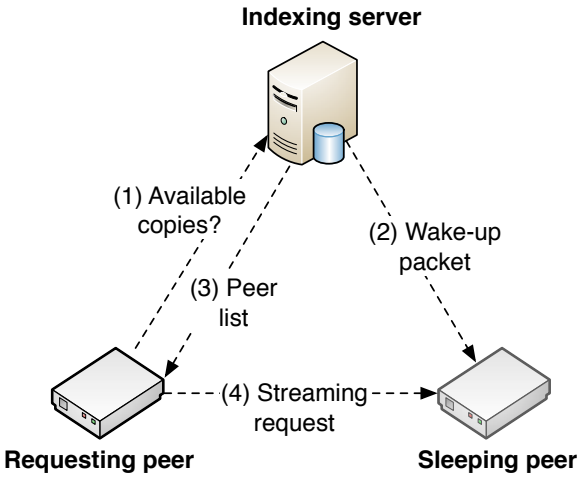


Figure 38: Example for *wake-on-demand* using Wake-on-LAN.

sired video and which of them should be woken up in order to provide the desired bandwidth.

Besides the knowledge about the potential STBs to wake-up, the wake-on-demand policy requires the ability to switch from the active or passive state into the sleeping state and to wake-up remote STBs on demand. While the former transition is trivial, the latter one deserves additional attention.

Wake-on-LAN

A possible way to wake up sleeping STBs is the usage of the Wake-on-LAN feature. This feature allows switching on a sleeping IP-enabled device from remote hosts by sending a so-called *magic packet* – directed LAN broadcast with the MAC address of the device and an optional 6-byte password [130]. The device to be woken up is sleeping with power reserved for the network card and, therefore, consuming only standby power (in the range of few watts [102]). Upon a reception of the magic packet, the network card verifies whether the contained MAC address matches its own MAC address, optionally checks the provided password, and, if the validation is successful, starts up the device. While the original Wake-on-LAN was designed for local (wired) networks only, it cannot be applied to remote connections over the Internet or to wireless networks. Therefore, different vendors and vendor consortia proposed several extensions to the original Wake-on-LAN specification [100, 69].

Figure 38 shows how the wake-up procedure can take place: the requesting peer sends a search query to the indexing server. The latter verifies whether sufficient non-sleeping STBs are available in the network. If it is not the case, the server sends a wake-up packet to selected sleeping STBs, and returns their IP addresses to the requesting peer, so that it can request video streams from the awakened peers.

Depending on the number of involved network domains, technology, and configuration utilized by the network operators, and the type of links (wired vs. wireless) several issues arise with this policy besides the required support of the Wake-on-LAN feature:

- Firewalls and routers might drop wake-up packets as a potential security threat. This introduces the requirement of custom configuration of the intermediate routers and Internet access gateways. Alternatively, Wake-on-LAN proxies could be used to bypass this issue but again require the existence of an always-on device in the network [125]. It follows that this issue can be solved if the overlay

provider also controls the network and users' gateways (as is the case where overlay provider is the network operator offering VoD services in its own domain or when overlay provider and network operator have a special agreement) but is not applicable in general.

- STB's wake-up time might delay the video startup for the requester by tens of seconds. Additional *patch streams* from the content server can alleviate this issue and avoid too long video startup times. The costs of the additional server traffic depend on the content being served in the system, i.e. it is relatively small for long videos and relatively large for short clips or trailers.
- The policy cannot be applied if the STB itself, such as DSL modems, must be running for Internet connectivity, since the Wake-on-LAN feature requires the sleeping STB to be reachable over the Internet. This limits the application of this policy to devices other than Internet gateways.

The discussed issues make the application of this policy limited or even impossible in certain scenarios, such as DSL modems acting as STBs or missing cooperation between the network and overlay provider. Therefore, we need an alternative policy that is able to approximate this behavior without relying on the Wake-on-LAN feature. Such a policy would enable STBs to save energy without the additional overhead of Wake-on-LAN proxies, patch streams etc.

6.5.2 *Supply-aware Recently Used Policy*

In order to overcome the restrictions of the previous solution, we propose an alternative policy that does not rely on the Wake-on-LAN feature. Instead, it requires only the common *timer* functionality, where an STB can switch into the sleeping state for the pre-defined amount of time. Such timers are common for DVRs, digital receivers etc., and can be incorporated into other kinds of STBs without the security limitations and wake-up delays of the wake-on-demand policy.

The idea behind this policy, called Supply-Aware Recently Used (SARU), is to make the available data match the estimated demand of the system by letting STBs stay online as long as their resources (cached content and upload bandwidth) might be needed by the system (see Figure 39). To achieve this, an STB analyzes the global availability of its cached content in regular intervals, both in the idle and in the sleeping state. In the latter case, the timer wakes up the STB for this examination. The decision whether to stay online or not is governed by a heuristic that tries to mimic the optimal behavior. By being adaptive to the demand and supply of cached videos, the heuristic tries to avoid STBs being unnecessary idle.

Note that this approach avoids the issues of the optimal policy by eliminating the startup delay (or patch streams from content servers) when a peer must be woken up remotely. Furthermore, the policy is also applicable to Internet gateways. Finally, SARU does not share the security-related problems of Wake-on-LAN over the Internet.

Before explaining the decision metric to switch nodes into the sleeping state and back (shown with dashed arrows in Figure 39), we define the system parameters:

- P: set of peers,
- V: set of videos,
- $V(p) \subseteq V$: set of videos cached at peer p,
- $P(v)$: set of peers that cached the video v, i.e. $P(v) = \{p \in P | v \in V(p)\}$,
- $R : V \to \mathbb{R}$: bitrate of a video $v \in V$,
- $u : P \to \mathbb{R}$: upload bandwidth of a peer $p \in P$,

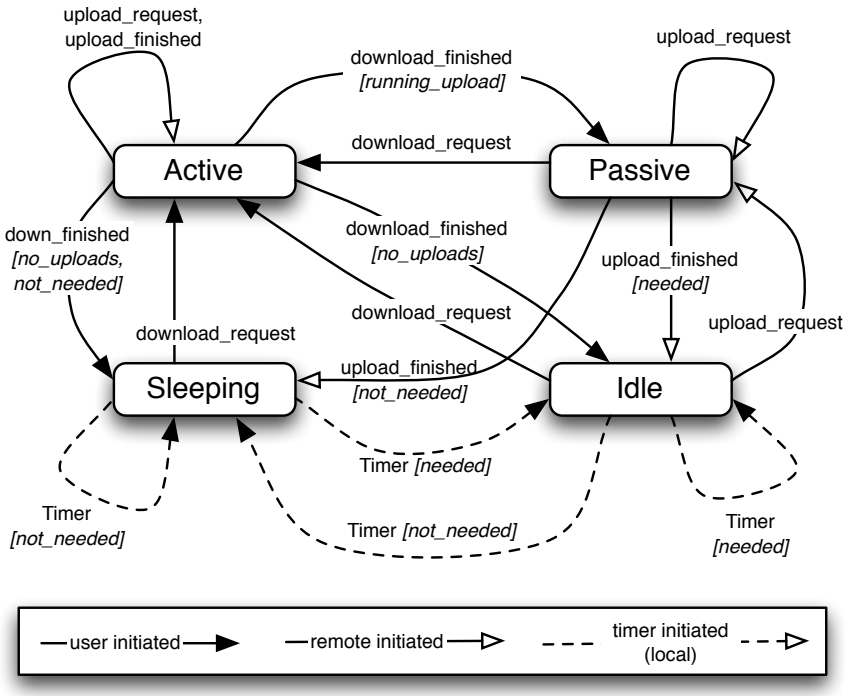


Figure 39: State transition diagram of SARU. *needed* and *not_needed*) specify the result of desirability calculations.

- $u' : P \to \mathbb{R}$: unused upload bandwidth of peer p with $u'(p) \leqslant u(p)$,
- $\phi : p \to \{0, 1\}$: whether peer p is *not* in the *sleeping* state.

In this regard, we define the total *availability* $\psi : V \to \mathbb{R}$ of a video v as:

$$\psi(v) = \sum_{p \in P(v)} \phi(p) \cdot u'(p). \tag{6.1}$$

and, accordingly, the availability $\psi'(v, p)$ that excludes the bandwidth of peer p:

$$\psi'(v, p) = \sum_{p' \in P(v) \setminus \{p\}} \phi(p') \cdot u'(p') = \psi(v) - u'(p). \tag{6.2}$$

Now let us assume that within the time interval τ, for a certain video v there will be n *additional* streaming requests. The required upload bandwidth to serve these peers in parallel is then $n \cdot R(v)$. Furthermore, let us assume that the average upload bandwidth of these peers is $\overline{u}$ and that all peers besides the last one could forward the received content. Then the currently available upload bandwidth for the given video will increase with the resources of new peers by $\overline{u} \cdot (n - 1)$.

Considering a peer p that has the given video in its local cache and decides whether to stay in the *idle* state and offer the content or to switch into the *sleeping* state, the resulting *missing upload bandwidth* δ that must be provided by the server to avoid playback degradations is (if p stays online):

$$\delta(v, n) = \max(n \cdot R(v) - \psi(v) - \overline{u} \cdot (n - 1), 0). \tag{6.3}$$

On the other hand, if peer p decides to switch into the sleeping mode, this results in the modified *missing upload bandwidth* δ' with:

$$\delta'(v, n) = \max(n \cdot R(v) - \psi'(v, p) - \overline{u} \cdot (n - 1), 0). \tag{6.4}$$

Since the goal of a standby policy is to reduce the probability that servers have to provide this bandwidth, having a perfect prediction of n would allow peer p to decide which decision to make. The metric would be to go to sleep if δ' is larger than zero and to stay online otherwise.

Alternatively, if n cannot be forecast with sufficient certainty, we propose to apply a *supply threshold* $\alpha \cdot R(v)$. That is, peers caching a specific video should stay online and provide the required bandwidth that will be used to stream this video to requesting peers if $\psi'(v,p)$ is below $\alpha \cdot R(v)$. The performance of this heuristic depends on the choice of the supply factor α and the *recheck interval* τ governing how often a peer evaluates the current situation.

To avoid that peers stay online and offer videos that are no longer popular (but are still in peer's cache), we additionally ignore videos that have not been requested within the last T hours. For this purpose, we define a *historical cutoff function* that specifies whether video v was requested by any peer (except p itself) within the last T hours or not:

$$H_T(v,p) = \begin{cases} 1 & \text{if } \exists p' \in P \text{ with last request from } p' \text{ for } v \text{ within } T,\ p' \neq p \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

The resulting *desirability* of each cached video v can then be calculated as:

$$\Delta(p,v) = \max\Big(\alpha \cdot R(v) - \psi'(v,p), 0\Big) \cdot H_T(v,p). \tag{6.6}$$

Consistently, the desirability of a peer p from the system's perspective is the aggregated desirability of all videos cached by peer p:

$$\Delta(p) = \sum_{v \in V(p)} \Delta(p,v). \tag{6.7}$$

Intuitively, the desirability metric reflects whether the locally cached files are sufficiently replicated in the overlay to satisfy additional streaming requests.

Only if the desirability $\Delta(p)$, calculated accordingly to Equation 6.7, is equal to zero peer p switches into the sleeping state and starts the *desirability recalculation timer*, which will wake up the STB after the recheck interval τ. However, if $\Delta(p) > 0$, p stays online and again starts the timer, which will initiate a new calculation of STB's desirability after the recheck interval τ (see Figure 39).

The length of this interval should be selected in such a way, that STBs do not wake up too often, since it will consume additional energy, and to avoid unnecessary communication overhead to calculate the desirability of cached content. Nonetheless, too seldom calculations might lead to wrong desirability estimation and result a high number of offline misses. For these reasons, we set the default value of the interval τ to one hour.

A proper configuration of the parameters T and α should allow SARU to achieve performance close to the wake-on-demand policy and outperform alternative policies. For example, a too low supply factor α might result in high load at content servers, while higher values of α will increase the online time of STBs. Furthermore, the history length T is important to react on content popularity changes. If T is chosen too short, not enough caching STBs will stay online to offer videos that are temporarily unpopular. However, if T is chosen too long, many STBs might stay online to offer quite unpopular content.

It is worth mentioning that the algorithm presented in this section works from the perspective of a single STB. Each STB makes individual decisions depending on the *content of its own cache* and *global* information about the popularity and supply of this content. In the following, we explain how this information is collected and managed.

6.5.3 *Reporting Overhead*

Both wake-on-demand and SARU require the knowledge about the current availability of each video in the overlay. The wake-on-demand policy must know how much bandwidth is currently missing for the requested videos, which offline peers possess a replica of this video, and, finally, how much bandwidth each peer can provide. Based on this information the peers to be woken up are selected. Similarly, SARU requires the information about the currently available bandwidth for certain videos (to perform the calculations described by Equation 6.6 and Equation 6.7).

There are two alternatives how to manage the information about the available bandwidth per video:

REACTIVE: The requesting peer asks each online replica owner for the available bandwidth. To this end, the indexing server only manages the information about all peers that downloaded the video in the past. Try-and-error method must be used to determine how much bandwidth the online replica owners can provide. Also the offline peers must be woken up to determine whether they still have a replica in their cache.

PROACTIVE: Peers report all relevant information to the indexing server, which manages an up-to-date information about available peers. Upon a peer's request (either download request or availability calculation required for the desirability calculation) this information is simply retrieved from the indexing server.

A simple estimation of the overhead required to reactively collect the availability information can be estimated as $O(|P| \cdot |V(p)| \cdot |P(v)| \cdot \frac{1}{\tau})$, since each of P peers has to check each $\frac{1}{\tau}$ minutes the availability of all locally cached videos $V(p)$ by asking all replica owners $P(v)$. This is quite inefficient with large caches and especially for highly popular (and therefore highly replicated) videos with $|V(p)| >> 1$ and $|P(v)| >> 1$. Another alternative, is an increase of the recheck interval τ, which might result in imprecise desirability calculation. Also for the Wake-on-demand policy, the reactive metadata management might result in unnecessary waking-up of STBs that don't possess a replica of the video anymore (due to the limited cache sizes).

Because of the presented considerations, we opt for the proactive management of available bandwidth per video, both for the online and sleeping STBs, which includes the per-video data about:

1. Lists of STBs that have a replica of the video in their cache.
2. Current state of each STB (online or sleeping), maximum available bandwidth (relevant for sleeping STBs), and currently available bandwidth (for online STBs only).

The calculation of this data requires that each STB reports to the indexing server the following events: (1) switches between the online and offline modes, (2) removal of videos from local caches or adding new videos to the cache, (3) changes in the currently available bandwidth caused by started and finished uploads. Since all of these actions happen only rarely (only few times per day for each STB) and the size of reported data is small (STB and video ids plus the available bandwidth), the overhead of such reports is limited, as presented in the evaluation section.

The information can be managed in various ways: by central indexing server, in a decentralized overlay (such as a Distributed Hash Table [139]), or via gossiping by piggybacking on already exchanged messages Since our system already contains centralized entities (content and indexing servers) we assign this responsibility to the indexing server. This solution allows us to reuse the normal search queries for available video sources to collect information required for the desirability calculations.

6.5.4 *Locality-aware Extension*

The presented heuristic policy, SARU, focuses on content supply in the whole system. From the network provider's perspective this might be suboptimal, since the content might be available in one part of the network but requested from remote network domains and, therefore, unnecessary inter-domain traffic might occur. Even the locality-aware peer selection (as described in Section 3.4.4) might not suffice here, since it requires the availability of *enough alternative content sources*, which, however, are reduced by the standby policy.

We propose the following modification to our heuristic: instead of considering the video availability in the *global system* (cf. Equation 6.1) we consider its availability in the local network domain by filtering out remote peers. The outcome is the Supply- and Locality-Aware Recently Used (SLARU) policy.

At first we define the metric whether two peers p and p' are in the same network domain:

$$\kappa(p, p') = \begin{cases} 1 & \text{if} \quad \mathrm{domain}(p) = \mathrm{domain}(p'), \\ 0 & \text{otherwise.} \end{cases} \tag{6.8}$$

Then we define the locality-aware extension of the availability metric ψ'(cf. Equation 6.2):

$$\psi'_\kappa(v, p) = \sum_{p' \in P(v) \setminus \{p\}} \phi(p') \cdot \kappa(p, p') \cdot u'(p') \tag{6.9}$$

so that the desirability metric can be computed by replacing ψ' with ψ'_κ in Equation 6.6.

It is straightforward that $\psi' \geqslant \psi'_\kappa$ is always true and, therefore, in a system that operates in multiple network domains, locality awareness comes at the cost of potentially higher online times. We evaluate this effect and its interplay with peer selection policies in detail in Section 6.7.4.

Table 12: Expected performance of considered policies.

Policy	Server traffic	Idle times	Remarks
Always-on	minimal	high	–
Selfish	high	minimal	–
Overtime	moderate	moderate	Depends on seeding time and request pattern
Wake-on-Demand	minimal	optimal	Requires Wake-on-LAN
SARU	moderate	near-optimal	Requires wake-up timer

6.6 EVALUATION METHODOLOGY

This section presents our evaluation methodology, including the workload, simulation environment, and metrics. The evaluation results are shown in Section 6.7.

Our expectations with respect to the performance of single policies are summarized in Table 12. The main subject of our evaluation is the heuristic SARU policy (as described in Section 6.5.2). We aim to understand, whether it can compete with the optimal Wake-on-demand policy and outperform the non-adaptive policies: always-on, selfish, and overtime, with respect to the system-wide performance and energy costs. Additionally, we consider the interplay of SARU and locality awareness techniques.

Table 13: Overview of the workloads generated by random sampling of original traces (and optionally geolocated to Germany).

	Workload	
Property	International	National (DE)
Video size	211 – 2,141 MB	211 – 2,131 MB
Number of peers	60,000	61,371
Number of videos	2,620	2,497
Number of downloads	118,610	101,609
Number of network domains	2,331	263
Total video volume	99,374 GB	87,256 GB

To capture the important effects of standby policies, we must consider a system-wide performance, not limited to a single video, but rather a peer-assisted system with multi-video caching on peers. This *scalability* requirement suggests the usage of simulations with an appropriate level of abstraction.

6.6.1 *Workload*

Realistic workloads are crucial to study the potential of standby policies. The workload must include realistic video popularity and request patterns such as the diurnal effect. Since there are no publicly available traces of video-on-demand systems suitable for our evaluation, we use traces of the video files being shared within the popular BitTorrent network.

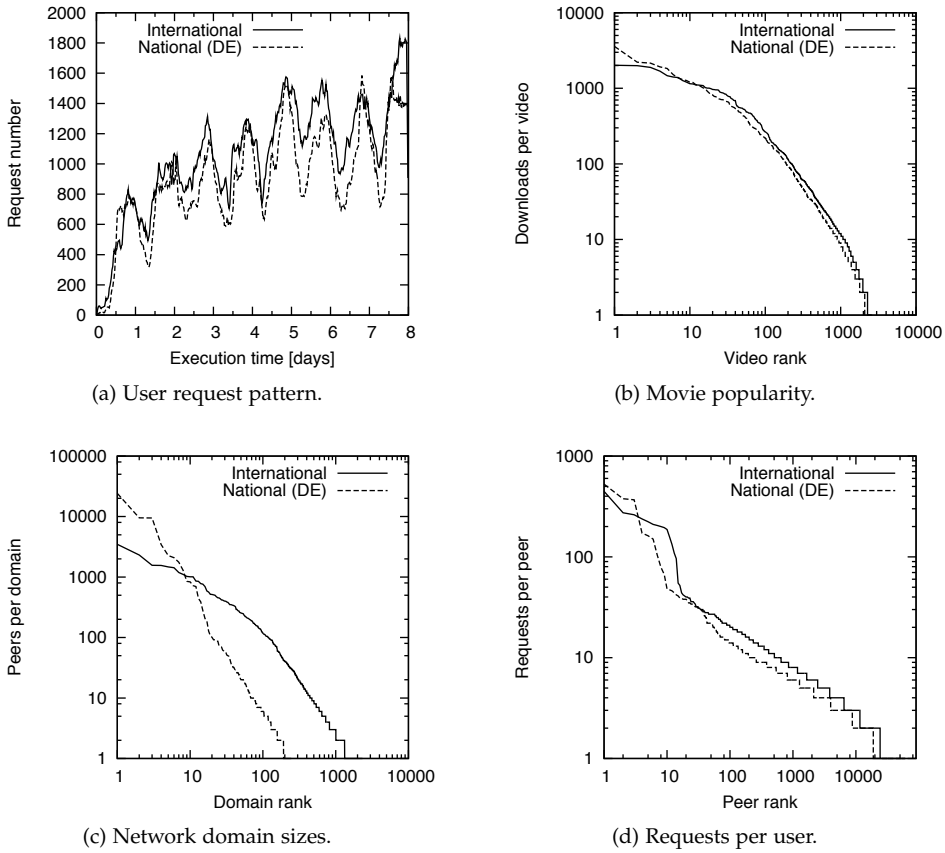
The traces were collected from December 9th, 2008 to January 16th, 2009 by subscribing to public RSS feeds of a major BitTorrent tracker site and then continuously asking the trackers for new peer lists [82]. The collected data includes the torrent name and id (used as a video identifier), the file size in MB, and the list of IP addresses active within the swarm at the given time. Based on this data, we can reproduce the time when peers requested certain videos. Furthermore, the peers are mapped to their respective autonomous systems by using the MaxMind's geolocation database GeoIP [101].

We generated two excerpts of the traces, by including only peers that downloaded content from the category *Movies* during the first week of measurements (Dec. 13-20, 2008), as shown in Table 13. For the *national* workload, all peers located outside of Germany were removed, which resulted in 61,137 peers. For the international workload, we took a random sample of 60,000 peers.

To understand whether the obtained traces are suitable to describe a video-on-demand scenario we also plot the relevant distributions in Figure 40. It can be seen that the workload shows a strong similarity with typical VoD workloads. Figure 40a presents the requests over time, which follow a clear diurnal patter (similar, for example, to the measurements reported by Huang et al. in [68, Figure 7]). Furthermore, the video popularity shown in Figure 40b follows the stretched exponential distribution of typical multimedia workloads [53]. Finally, we observe that the user activity in terms of the video requests is highly skewed (see Figure 40d).

We further observe that the number of network domains and the number of peers per domain differ between the national and international workloads (see Figure 40c): For Germany, the same amount of peers is distributed over less domains but the large domains contain more peers than with international peers.

Table 14 shows further parameters of the basic scenario used for the performance evaluation. We use 8 continuous days out of our traces to generate each of the work-



(a) User request pattern.

(b) Movie popularity.

(c) Network domain sizes.

(d) Requests per user.

Figure 40: Properties of trace content.

loads. In order to verify the significance of our results, we repeated experiments with six different weeks. Since the outcome is consistent among the weeks, we present the results of a single week. The minimal size of considered videos is 200 MB and we derive the bitrate of the video R depending on the video size S according to the following rule:

$$R = \begin{cases} 1\text{Mbps} & \text{if} \quad S \leqslant 1\text{ GB} \\ 2\text{Mbps} & \text{if} \quad 1\text{ GB} < S \leqslant 2\text{ GB} \\ 4\text{Mbps} & \text{if} \quad S \geqslant 2\text{ GB}. \end{cases} \tag{6.10}$$

The default peer upload bandwidth is set to 500 kbps, while the upload bandwidth of the server is unlimited to allow for measuring the server stress. We use a sample of up to 60,000 random peers from our traces (consisting of 4 million peers for the selected content category *Movies* for the considered week). Table 13 summarizes the major properties of the both resulting workloads.

The traces are fed into a custom discrete event-driven simulator modeling a peer-assisted streaming system based on STBs. We implemented the delivery network presented in Section 6.3 with the users consuming videos and overlay provider's servers offering the initial content by extending the simulator described in Section 5.6. The overlay peers model the STBs, including limited-sized local caches, trace-driven requests, and video popularity, as well as the standby policies described in Section 6.5.

The data transfers happen in a multi-source manner, which means that a single streaming request can be served from multiple nodes (peers and/or servers), while peers are always favored in order to offload servers.

Table 14: Basic setup.

Parameter	Default	Variation
Simulation duration	8 days	–
Video bitrate	1, 2, or 4 Mbps	–
Peer cache size	2 GB	–
Cache replacement policy	FIFO	–
Peer selection policy	Upload volume	locality-aware
Peer upload bandwidth	0.5 Mbps	0.250 – 4 Mbps
Number of peers	60,000	10,000 – 60,000
Number of videos	2,620	1,260 – 2,620
Workload	international	national (DE)
Standby policy	SARU	always-on, selfish, overtime, wake-on-demand

6.6.2 *Power Consumption Model*

In order to assess the power consumption of devices, we consider three different categories and take a representative device for each of them. We distinguish four different states specified in Section 6.4: sleeping, idle, active, and passive.

The resulting consumption profiles are shown in Table 15. Here, *home gateways* are represented by Thomson TriplePlay VDSL2+ modem, whose management consumption was evaluated and reported by Valancius et al.[148]. Classical *set-top boxes* are represented by Motorola's VIP1216, an IP-enabled DVR equipped with a hard disk drive. Its power consumption was measured by the ENERGY STAR consortium [42]. Finally, the latest generation of *game consoles* is represented by Sony's PlayStation 3 Slim device which is already 50% more efficient than its predecessor [107]. This differentiation of device types is relevant in order to understand which of the considered policies are feasible for which device type.

Table 15: Power consumption profiles of various STB types (based on [42, 148, 107]).

Device type	Energy Consumption [Watt]			
	Sleeping	Idle	Passive	Active
Home gateway	0.50	14.90	15.40	16.75
Set-top Box (DVR)	0.50	10.78	12.00	16.00
Game console	0.36	75.19	80.90	80.90

6.6.3 *Metrics*

In order to assess a standby policy π, we measure $\Gamma_p(\pi)$ and $\Gamma_s(\pi)$ as the data volume uploaded from peers and servers, respectively, and $\Omega(\pi)$ as the total online time of all STBs. Based on these values we calculate the following metrics (using the always-on policy A as a reference):

- *Relative server traffic* γ is the data volume uploaded from content servers relative to the totally served data:

$$\gamma(\pi) = \frac{\Gamma_s(\pi)}{\Gamma_s(\pi) + \Gamma_p(\pi)}.$$

  This metric describes the *performance* of the system, where the lower server traffic means higher benefit for the overlay provider.

- *Relative online time* ω is the accumulated online time of all STBs relative to the always-on case:

$$\omega(\pi) = \frac{\Omega(\pi)}{\Omega(A)}.$$

  This metric describes the users' *costs* that should be reduced by the standby policies (we assume that the users pay flat-rate fees for Internet access and that the baseline power consumption dominates the energy costs of STBs, similar to [148]).

- *Energy savings* ρ express the relative reduction in STBs' online time achieved by policy π compared to the always-on case. We further normalize this value by the server traffic reduction achieved by π compared to the always-on policy A for a fair comparison. The resulting normalized energy savings are:

$$\rho(\pi) = \frac{\Omega(A) - \Omega(\pi)}{\Omega(A)} \cdot \frac{\Gamma_p(\pi)}{\Gamma_p(A)}.$$

- *Messaging overhead* measures the number of messages exchanged between the STBs and the indexing server per second. This includes search messages when a user requests a video download, state changes reported by STBs when they switch into the online or offline states, and upload capacity changes when STBs report that they start or stop new upload streams. This also includes two types of messages specific to the wake-on-demand policy and SARU: wake-up messages and control lookups required to calculate the desirability of an STB.

6.7 EVALUATION RESULTS

In this section, we present the performance results of our adaptive heuristic policy and compare it with the performance of alternative policies. The results were obtained using trace-driven simulations according to the methodology presented in the previous section.

According to our evaluation goals, the following experiments were conducted:

COMPARISON OF STANDBY POLICIES covers the performance of all considered policies with the default setup (with the international workload). The main goal is to understand the relationship of considered metrics and to understand whether the proposed adaptive standby policies can outperform static policies.

ENERGY CONSUMPTION VS. DEVICE PROFILE compares the impact of different policies on the energy consumption depending on the device type in use. This allows to understand which policies are suitable for STBs, home gateways, or game consoles.

ANALYSIS OF THE SUPPLY-AWARE RECENTLY USED POLICY provides a parameter sensitivity study that considers all three relevant parameters of SARU and their impact on the SARU's performance and costs.

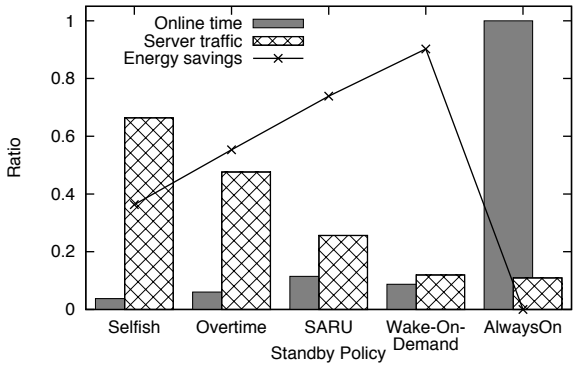


Figure 41: Relative online time and resulting server traffic of various policies.

Locality Awareness and Standby Policy analyzes SARU's effects on the inter-domain traffic under different locality-aware mechanisms. In this case, we also include the results of the national workload to highlight the effect of internationalization on the costs and gains of locality-aware mechanisms.

Additional evaluation results can be found in Section A.3, including Scalability of Standby Policies, Performance with a National Workload, and Additional Results for Locality Awareness.

6.7.1 *Comparison of Standby Policies*

In this experiment, we compare the performance of the proposed heuristic SARU policy with that of other standby policies: always-on, wake-on-demand, overtime, and selfish. Thereby, we consider the *international workload* (see Section 6.6.1), while the results for the national workload are presented in Section A.3.2 for the purpose of completeness.

The main goal of this experiment is to verify our expectations with respect to the single policies listed in Table 12. Thereby, we set the seeding time of the overtime policy to one hour and configure SARU as follows: recheck interval τ equals one hour, history length T equals 24 hours, and supply factor α is set to 1.0.

The considered metrics are the ratio of the total traffic served by servers and the online time of peers. Figure 41 shows the results shown and provides a direct comparison of considered standby policies by including the normalized energy savings.

First, we observe that the selfish and always-on policies result in the lowest (3.75%) and highest (100%) online times, respectively. Second, the selfish policy generates the highest server traffic of 64%. Furthermore, the always-on and wake-on-demand policies offer the lowest server traffic (11 and 12%) accounting for the initial content injection from the content servers.

We observe that our heuristic SARU policy yields server traffic closer to the optimal wake-on-demand policy (26 vs. 12%) while slightly increasing the online time (11.4 vs. 8.7%). Thereby, *SARU clearly outperforms the selfish and overtime policies with respect to the server traffic, and the always-on policy with respect to the online time metric*.

Finally, with respect to the normalized energy savings metric, we again observe that SARU outperforms selfish, overtime, and always-on policies, but ranges below the wake-on-demand policy.

These results confirm our expectations that the wake-on-demand policy offers optimal trade-off of online time and server traffic, while SARU approximates this behavior.

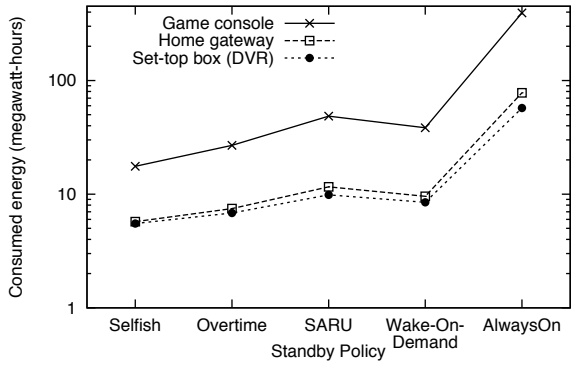


Figure 42: Power consumption per device with the international workload (y axis is shown on a logarithmic scale).

6.7.2 *Energy Consumption vs. Device Profile*

In this experiment, we aim to understand the applicability of considered policies to different types of STB-like devices: home gateways, DVRs, and game consoles. The experimental setup reuses the configuration of the previous experiment but also measures the power consumption of respective devices in different states (as specified in Table 15).

Figure 42 shows the total energy consumption of all 60,000 peers over the duration of 8 days (though not all peers joined the network from the start). It is further important that certain combinations of device types and policies are rather hypothetical (for example, it is difficult to implement Wake-on-LAN on home gateways because this policy requires an existing Internet connection).

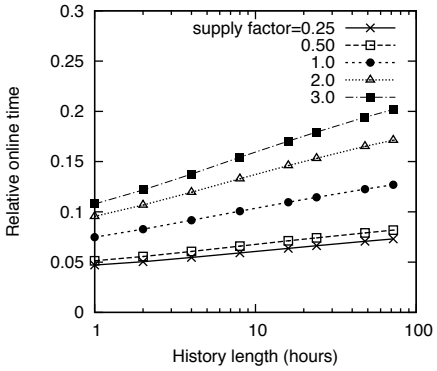
The outcome of this experiment confirms our expectation that especially game consoles can significantly benefit from the adaptive policies, since the overall energy consumption can be reduced by the factor eight. In this regard, especially wake-on-demand enables considerable energy savings, which is only 38 MegaWatt hour (MWh) compared to 48 MWh of the SARU policy, or ~400 MWh of the always-on policy. While the overtime policy consumes with 20 MWh less energy than SARU, it also results in a higher server traffic as shown in the previous experiment.

For the other devices the wake-on-demand policy increases the energy consumption moderately, but the (relative) difference to other policies is much lower. For example, with DVRs we observe 8.5, 9.8, and 6.8 MWh for the wake-on-demand, SARU, and overtime policies, respectively. However, even here, each of these policies consumes five to eight times less energy than the always-on behavior.

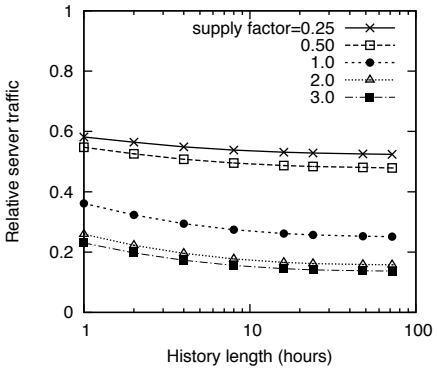
To conclude this analysis, standby policies should be used especially in the context of high-end devices, such as game consoles, while the other two device types can benefit from it. Furthermore, the obtained results justify our focus on normalized energy savings as the main metric to assess the performance of standby policies in a device-independent way.

6.7.3 *Analysis of the Supply-aware Recently Used Policy*

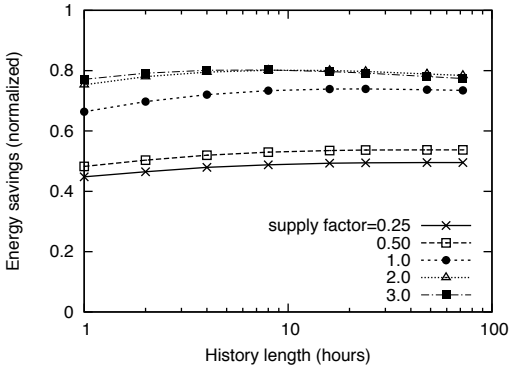
In the previous experiments the parameters of the SARU policies were fixed. In this experiment, we present a parameter study of the SARU policy and observe the impact of single parameters on the policy's performance and the incurred overhead.



(a) Relative online time.



(b) Relative server traffic.



(c) Energy savings (normalized).

Figure 43: Impact of *history length* and *supply factor* on the SARU's performance (x axis is shown on a logarithmic scale).

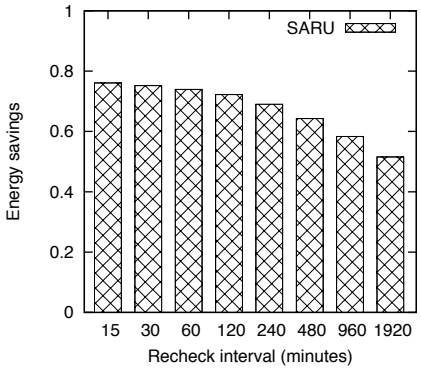
Impact of Supply Factor and History Length

We start the parameter sensitivity study by varying the following parameters: *history length* T in the range from 0 to 72 hours and *supply factor* α from 0.25 to 3.0. The results are shown in Figure 43 and include the relative server traffic, relative online time, and normalized energy savings.

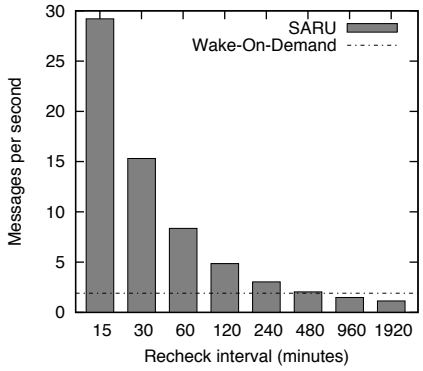
We observe that the online time slightly increases with the growing history length while even with the increased supply factor the total fraction is below 20% (see Figure 43a). Furthermore, the relative server traffic can be reduced significantly from 60 to 14% by increasing the supply factor to 3.0 and setting the history length to at least 24 hours (see Figure 43b). The reason is that a longer history makes peers stay online longer to offer their content, while the supply factor $\alpha \geqslant 1.0$ allows serving at least one new request by the peer kept online.

Since with the increasing parameter values the energy costs (in terms of the total online time) and the server traffic is reduced simultaneously, we also consider the normalized energy savings (shown in Figure 43c). Furthermore, the relative energy savings increase with the history length and higher supply factor but stabilize at the supply factor of 2.0 and history length of eight hours (see Figure 43c). Thus, the supply factor of 2.0 and history length of roughly eight hours result in the highest savings. Also the increase of the supply factor from 2.0 to 3.0 only results in marginal additional savings with a short history of less than two hours. These observations suggest that a further increase of the supply factor and history length is not beneficial.

Finally, we also measured the impact of supply factor and history length on the messaging overhead but (as expected) did not discover a significant impact. Therefore,



(a) Energy savings (normalized).



(b) Messaging overhead.

Figure 44: Impact of *recheck interval* on the SARU's performance and messaging overhead (x axis is shown on a logarithmic scale).

we present this metric only in the subsequent experiment where an impact can be observed.

Impact of Recheck Interval and Messaging Overhead

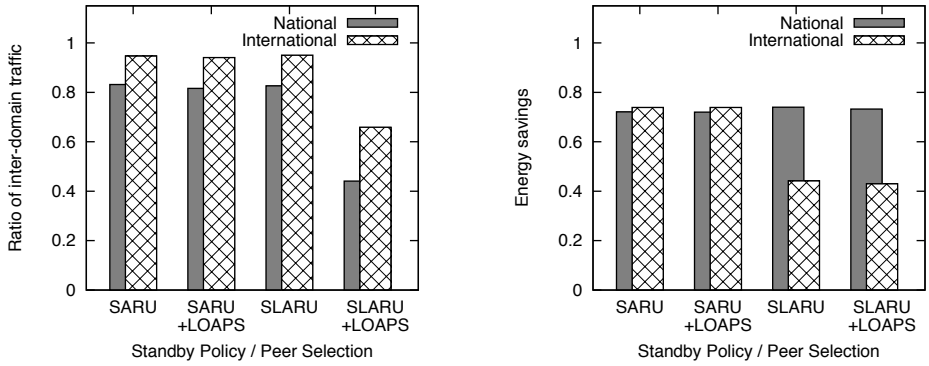
In this experiment we vary the remaining parameter of the SARU policy to evaluate its impact on the energy savings and messaging overhead. Especially, the latter metric is relevant since we expect that too frequent wake-up timer events might result in too many desirability calculations and, accordingly, in a high load at the indexing server. For this purpose, we set the other two parameters to their initial values ($\alpha = 1.0$ and $T = 24$ hours) and vary the recheck interval from 15 minutes to 32 hours (=1920 minutes) by doubling its value in each step. The obtained results are presented in Figure 44.

Our first observation is that the normalized energy savings decrease only slightly with the increasing recheck interval (see Figure 44a). Even an interval of one or two hours still results in considerable savings of 74 or 72%, respectively. Furthermore, the messaging overhead results reveal that status events indeed cause most of the messaging overhead at the indexing server. Indeed, the overhead reaches the level of the wake-on-demand policy only with the recheck interval of eight hours (=480 minutes).

6.7.4 *Locality Awareness and Standby Policy*

So far we considered the performance of standby policies only from the users' and overlay provider's perspectives, who aim to reduce energy consumption of STBs without incurring high load on content servers. In contrary in this experiment we analyze the heuristic SARU policy also from the perspective of network operators. Similar to Chapter 5, we consider the inter-domain traffic as the main cost factor for the network operators, since it requires well-dimensioned inter-domain links and might additionally result in high interconnection payments.

Since the basic version of SARU does not consider any information about the underlying networks. This might result in lower availability of content in the individual network domain and, therefore, leads to higher inter-domain costs. In Section 3.4.4 and Section 6.5.4 we already presented two mechanisms, LOcality-Aware Peer Selection (LOAPS) and Supply- and Locality-Aware Recently Used (SLARU) that might be able to alleviate the network operator's situation. Both mechanisms can be applied independently and, therefore, result in four possible combinations: (1) pure SARU (with-



(a) Ratio of inter-domain traffic.

(b) Energy savings (normalized).

Figure 45: Impact of locality awareness with national and international workload (SLARU = locality-aware SARU, LOAPS = locality-aware peer selection).

out LOAPS), (2) SLARU (without LOAPS), (3) SARU with LOAPS, and (4) SLARU with LOAPS.

An additional factor is the workload-specific distribution of users on operator's domains. To understand its impact on the inter-domain traffic and the performance of the aforementioned mechanisms, this time we consider two variants of our trace-based workload: national (located within Germany) and international subset of peers, as described in Section 6.6.1.

The performance of considered locality mechanisms for both workloads is shown in Figure 45, presenting both the ratio of inter-domain traffic and the normalized energy savings achieved by each combination of mechanisms. We make the following observations with respect to the inter-domain traffic (see Figure 45a):

1. For each combination of mechanisms the *national* workload results in lower percentage of inter-domain traffic than the corresponding international workload (10-40% less). This can be easily explained, since the national workload contains less network domains and has more users per domain, which, in turn, results in higher percentage of *local sources* for each download request (see Figure 40c).

2. SLARU by itself is not able to significantly decrease the percentage of inter-domain traffic (compared to pure SARU). Even if the number of local online peers is increased, the probability that local sources are selected for the download is still low. In fact, this probability is *inversely proportional* to the number of network domains, which is quite high for both setups (263 and 2,331 domains).

3. LOAPS alone has almost no effect with pure SARU, which can be explained by the low number of potential download sources in individual network domains and, therefore, low effect of selection mechanisms in use.

4. Combination of SLARU with LOAPS achieves significant reduction in inter-domain traffic: 30% for the international and almost 50% for the national workload. Here, SLARU assures the local supply of content, while LOAPS assures the selection of local download sources. The reason of better performance with the national workload, is again the lower number of network operator domains.

Concerning the normalized energy savings achieved by the standby policy in various setups, we observe (see Figure 45b):

1. LOAPS alone has no effect on energy savings, independently of the workload type and SARU variant in use. The reason is simply that peer selection has no direct impact on the state switches performed by SARU.

2. Usage of SLARU (with or without LOAPS), has almost no impact for the national workload, because most of the peers are located inside of few large network domains. Though, the online time increases, the corresponding reduction in server traffic results in *efficient* utilization of online times and, therefore, the normalized energy savings remain stable or even increase slightly. Detailed results for server traffic and online times are additionally provided in Section A.3.3.

3. With the international workload the energy savings are reduced from 74 to almost 40% if SLARU is applied. The reason is that the large number of network domains results in too many STBs staying online to guarantee the availability of content that may be never requested within the same network domain.

We conclude that the *locality-aware extension to SARU* (SLARU) makes sense only in combination with LOAPS, where it is able to reduce inter-domain traffic without hurting the overlay performance. Furthermore, for overlays which span large number of network domains, which is the case for the international workload, additional effort from the network operator is needed to incite locality-aware behavior in the overlay (cf. Chapter 5).

6.8 SUMMARY

This chapter presented adaptive standby policies that are able to reduce energy consumption of a peer-assisted content delivery network based on STBs. We achieved this by reducing the *idle* times and, therefore, energy consumption of STBs. The STBs can switch into the sleeping state and back in a self-managed way to save the baseline power. To achieve this, we devised two policies: Wake-on-Demand and Supply-Aware Recently Used (SARU) that rely either on the Wake-on-LAN feature or on wake-up timers to control the online state of STBs.

Our evaluation study showed that adaptive standby policies can tackle the trade-off between the extensive online time and low server traffic, resulting in a similar performance as the always-on behavior at slightly higher costs than the selfish policy. The proposed policies also outperformed the non-adaptive overtime behavior. For example, our *wake-on-demand* policy can reduce the server online time by up to 91% while providing the same performance (with respect to the server traffic) as the costly always-on policy. In cases when the wake-on-demand policy cannot be applied, the alternative policy, SARU, enabled comparable energy savings with a slightly lower performance.

Additionally, we addressed the trade-off between the energy savings for the users and inter-domain traffic reduction for network operators. On the one hand, a proper combination of locality-aware peer selection and locality-aware standby policy was able to significantly reduce inter-domain traffic. On the other hand, we saw that locality awareness by itself can be either neutral or even harmful for the overlay, depending on the number and relative size of involved network domains.

7 CONCLUSIONS

In this chapter, we summarize our main results and outline the contributions of this thesis. We further provide an outlook at possible future research directions.

7.1 SUMMARY

This thesis considered peer-assisted Video-on-Demand (VoD), which is a promising approach to offer large video collections to many users in a scalable and cost-efficient way. Due to its hybrid nature, a properly designed peer-assisted system is able to provide real streaming experience to the users, while significantly reducing the content delivery costs in comparison to server-based systems.

In Chapter 1, we introduced the *stakeholders* involved in peer-assisted VoD: overlay providers, users, and network operators. We discussed the challenges arising from the utilization of the peer-to-peer (p2p) paradigm: the limited peer resources, the peers' inability to provide service guarantees, and insufficient incentives in terms of peers' contribution. We argued that the design of a peer-assisted system must also account for the costs of end-users (such as the energy consumption of their devices) and for the impact on the underlying network infrastructure.

In Chapter 2, we presented the *background* of peer-assisted VoD. First, we discussed different delivery architectures: client-server, Content Delivery Networks (CDNs), pure p2p, and peer-assisted systems. We also compared them in terms of their benefits and limitations. Second, we highlighted the differences between various methods of video delivery: VoD streaming, live streaming, and file transfers. We further presented the mesh- and tree-based topologies for video streaming, including an example of a mesh-based p2p streaming protocol. Subsequently, we discussed the impact of Internet-based delivery overlays on the underlying network infrastructure. Such overlays generate a large amount of inter-domain traffic, which is undesired from the network operators perspective. Finally, we introduced the upcoming delivery platforms that utilize Set-top Boxes (STBs) and similar devices of end-users. On the one hand, these devices enable more control through the overlay provider. On the other hand, STBs require new management techniques to avoid wastage of peers' resources and resulting disincentives for peer-assistance.

Chapter 3 presented the peer-assisted *architecture* for VoD, which built the scenario considered in this thesis. The chapter further presented our assumptions and problem statement. The problem statement included four aims: guaranteed streaming experience for users, reduced load on content servers, reduced inter-connection costs for network operators, and, finally, reduced energy consumption of peers (in the STB-based scenario). Subsequently, we presented selected components of the considered architecture that are relevant in this regard. The components included the utilized streaming protocol, server utilization, search mechanism, and caching strategy (at peers).

Chapter 4 presented *adaptive resource allocation policies*. An overlay provider can apply these policies to its content servers to assure that server resources are used only to compensate for temporary unavailability of peer resources. Additionally, in an STB-based system, allocation policies can be used to determine when certain STBs can be switched off to save energy. To this end, we provided a model for the instantaneous bandwidth demand of such a system and proposed two resource allocation policies for peer-assisted streaming systems. Our first policy was inspired by the related work on p2p file sharing while the second policy addressed the specific requirements of streaming applications. We evaluated both policies in various scenarios via extensive

simulations and demonstrated their benefits (compared to each other and to a perfect static allocation). Thereby, our policies were able to provide high streaming quality (short startup delays and no stalling). At the same time, the load on content servers was reduced by 30–50% compared to a non-adaptive solution. This was achieved with only 1% of additional messaging overhead. We further presented the prototypical integration of the *supporter* policy in a state-of-the-art p2p streaming application. The measurements of the prototype in a testbed verified the correctness of the obtained simulation results.

In Chapter 5, we addressed the tension between the peer-assisted overlays and network operators. Our goal was to meet the diverse requirements of users, overlay providers, and network operators. A network operator must try to satisfy the quality requirements of its customers to attract them. At the same time, the operator aims to minimize the traffic costs, particularly those incurred from inter-domain traffic. In the case of commercial peer-assisted video delivery, also the interests of the overlay provider must be considered to avoid network neutrality violations. We discussed existing approaches that are able to reduce inter-domain traffic, which often degrade the overlay performance, fail to support various protocols, or introduce legal issues by being content-aware. Therefore, we devised an *incentive-based traffic management* mechanism. The proposed mechanism, called Highly Active Peer (HAP) promotion, decreases network operators interconnection costs by encouraging the overlay providers and users to be network-friendly. For this purpose, a network operator tries to detect most network- and overlay-friendly peers and boosts their performance by increasing their bandwidth for certain time intervals. Our mechanism enables a reduction of costly inter-domain traffic by up to 50% without punishing the users via traffic shaping or blocking. At the same time, we observed an increased content and bandwidth availability in the local domain, which allowed to reduce the load on content servers by up to 88%. We further analyzed the usefulness of this approach for single network operators. While the approach was always able to reduce the inbound traffic, an additional mechanism allowed also to avoid an increase in the outbound traffic as desired by certain network operators.

In Chapter 6, we considered the case of peer-assisted architectures built on STBs controlled by the overlay provider. Such devices are a promising alternative to desktop PCs to build peer-assisted delivery overlays. However, the overlay provider must take the costs of an STB owner into account, especially the energy consumption of STBs. In this regard, the related work typically assumes only extreme policies to control online time of STBs: they are either always-on or switched off if not actively used by their owners. While the first approach might result in wastage of the STBs' baseline power, the second approach increases the load on content servers. Therefore, we proposed two sophisticated *standby policies* to provide high performance while reducing the idle times by turning STBs offline in a self-managed way: the wake-on-demand policy and the heuristic Supply-Aware Recently Used (SARU) policy. Our policies rely on special features of STBs, which offer an interesting compromise between the extreme approaches. The evaluation studies demonstrated that the proposed policies are able to save up to 80% of the baseline power of dispensable STBs by sending them into a sleeping state. At the same time, they are able to offload servers at a level similar to the energy-wasting always-on case. We also measured the overhead of the standby-policies (in terms of signaling messages) and their effectiveness with different device profiles. Finally, the chapter also considered the trade-off between the network-friendliness and energy efficiency of a peer-assisted streaming overlay. We showed that a combination of a locality-aware extension to the SARU standby policy with locality-aware peer selection can reduce inter-domain traffic by 30–50% depending on the geographical distribution of the overlay.

7.2 CONTRIBUTIONS

This section outlines the contributions of this thesis towards the cost reduction and performance optimization of peer-assisted VoD. We start with the major contributions of this thesis:

1. *Adaptive cache allocation policies* that enable the overlay provider to avoid wastage of costly server resources. At the same time, the policies provide the desired level of streaming quality to the users. Our evaluation studies confirmed that, in particular, our *supporter* policy is able to satisfy the requirements of peer-assisted VoD. In contrast to the related work, our policies neither rely on predictions of the user demand (cf. [155, 111, 128]), nor they assign static resources to the system (cf. [48, 66, 22, 32]). Instead, our policies measure the current bandwidth demand and supply in order to allocate sufficient resources in an adaptive manner.

2. *An incentive-based traffic management approach* that allows network operators to manage peer-assisted overlays without hurting their performance. For this purpose, the mechanism incites the users and overlay provider to behave network-friendly, instead of forcing them. As a result, network-friendly behavior becomes a dominant strategy, both for the overlay provider and users. Due to its incentive-based nature, this mechanism avoids the common limitations of existing approaches such as the decreased overlay performance, applicability only to specific protocols (e.g. BitTorrent), content awareness, and violations of the network neutrality principle (e.g. [109, 117, 104, 17, 92, 33]).

3. *Adaptive standby policies for STBs* that aim to reduce the idle times of such devices while keeping a sufficient number of online copies. Existing approaches for STB-based systems often assume non-adaptive behavior, such as always-on or selfish [60, 74, 148, 27]. This behavior often results in energy wastage or too high load at the content servers. Therefore, we proposed two adaptive standby policies that utilize additional features of STBs such as Wake-on-LAN and wake-up timer. The trace-driven performance evaluation verified the desired properties of the proposed policies and their superiority compared to non-adaptive approaches.

There are further findings and insights that we obtained in this work. They constitute additional contributions of this thesis. We describe them briefly in the following:

- We provided a theoretical model for the instantaneous bandwidth demand of peer-assisted VoD. We used the model to devise the *global speed* allocation policy (see Section 4.4.1), while there exist further works relying on it [2].

- Our evaluation of proposed cache allocation policies revealed that the *supporter* policy that focuses on the playout buffers outperforms policies based on the download speed measurements (see Section 4.6).

- We implemented the *supporter* policy in a state-of-the-art p2p video streaming client. This prototype was used to verify simulation results in a testbed (see Section 4.7).

- We showed that our cache allocation policies can be applied not only to the allocation of server resources, but also to the resources of non-downloading peers, such as idle STBs. We observed that with STBs the server traffic reduction is higher than without STBs (see Section 4.6.4). Furthermore, this mechanism allows to determine the number of currently required STBs while the remaining devices can apply the standby policies.

- We analyzed the performance of the proposed standby policies for different types of STB-like devices: home gateways, Digital Video Recorders, and game consoles. The results revealed that especially game consoles can significantly reduce their energy consumption if they use our standby policies (see Section 6.7.2).
- We also demonstrated that a standby policy can take the underlying network infrastructure into account (see Section 6.7.4). For this purpose, we developed a network-friendly extension, called Supply- and Locality-Aware Recently Used (SLARU), that was able to reduce the inter-domain traffic by 30–50% if combined with the locality-aware peer selection.
- Our incentive-based traffic management mechanism aims to promote peers that are able to localize the overlay traffic. For this purpose, we considered basic metrics to select peers: contribution, network-friendliness, and seeding ratio. Our evaluation study revealed that the combination of the network-friendliness and contribution metrics enables an appropriate trade-off between the overlay performance and the required amount of inter-domain traffic (see Section 5.7.2).
- We also paid special attention to the *early adopter* case where only one network operator applies the proposed traffic management mechanism (see Section 5.7.3). The evaluation showed that: (1) large operators benefit more than small operators and (2) the early adopter decreases its *inbound* traffic but also increases its *outbound* traffic. To this end, we demonstrated how a restriction of the additional bandwidth to the local domain can avoid the outbound traffic increase.

7.3 OUTLOOK

The contributions of this thesis covered selected aspects of peer-assisted multimedia content delivery. Due to the complexity of peer-assisted systems, the interaction of proposed mechanisms with further components can influence the system's performance. Therefore, improvements of other components and better integration between them can additionally improve the performance and reliability of peer-assisted streaming [4, 99]. For example, the research community is currently actively investigating replication mechanisms, advanced coding techniques (such as scalable video coding), error correction methods, and incentive schemes for user participation [160, 4, 106, 81]. Since these approaches might also incorporate certain level of adaptivity, their interplay with the adaptive server allocation and adaptive standby policies appears interesting. Furthermore, a combination of scalable video coding and server allocation would require a decision whether a system should react to performance variations with changed server allocation or changed video quality level. Here, performance metrics related to the quality of experience can be used to make the right decisions [64, 165, 45, 3]. Similarly, a combination of adaptive standby policies with proactive content replication might require additional policy extensions but also result in even better performance.

Commercial peer-assisted video-on-demand was the main scenario of this thesis and, consistently, has been used in our evaluation. Nevertheless, the proposed mechanisms can be also applied to file transfers and live streaming, which is promising but might require minor mechanism modifications. For example, the incentive-based traffic management can be applied not only to peer-assisted but also to pure p2p systems. As a result, traffic management would affect the users' performance instead of the content servers' load. Additionally, deployment by network operators could provide interesting insights into the feasibility of this mechanism in real networks, such as currently performed within the SmoothIT project [136].

Incentives for user contribution is another highly relevant topic. Without incentives, a peer-assisted solution cannot be successful since most of the content would

be downloaded from servers. This thesis provided a mechanism to reduce the load on end-users devices achieved through adaptive allocation of (peer) caches. Furthermore, our adaptive standby policies improve the energy consumption of end-user devices. However, the exact agreement how the overlay provider rewards users for their contributions was out of scope of this thesis. Future work could assess in detail the required amount of rewards for user's contribution in a commercial scenario.

Another relevant trend is the development of the network infrastructure towards the Future Internet [146]. The increased intelligence of overlay applications and their demand for service-specific quality guarantees (such as video streaming in our scenario) require higher scalability, modularity, and flexibility inside of the network. Our approach to dynamic management of user bandwidth (through HAP promotion) is one possibility to improve the efficiency of Internet-based content delivery. Future approaches could add further service-specific intelligence in the network to enhance the performance and cost-efficiency of content delivery overlays.

[1] Osama Abboud, Aleksandra Kovacevic, Kalman Graffi, Konstantin Pussep, and Ralf Steinmetz. Underlay Awareness in P2P Systems: Techniques and Challenges. In *23th IEEE International Parallel and Distributed Processing Symposium, Workshops and Phd Forum (IPDPSW)*, 2009.

[2] Osama Abboud, Konstantin Pussep, Markus Mueller, Aleksandra Kovacevic, and Ralf Steinmetz. Advanced Prefetching and Upload Strategies for P2P Video-on-Demand. In *ACM Workshop on Advanced video streaming techniques for peer-to-peer networks and social networking*, 2010.

[3] Osama Abboud, Thomas Zinner, Konstantin Pussep, Simon Oechsner, Ralf Steinmetz, and Phuoc Tran-Gia. A QoE-Aware P2P Streaming System Using Scalable Video Coding. In *10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2010.

[4] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Enabling Resilient P2P Video Streaming: Survey and Analysis. *Multimedia System Journal (MMSJ)*, 17(2):1–21, 2011.

[5] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can ISPS and P2P Users Cooperate for Improved Performance? *ACM SIGCOMM Computer Communication Review*, 37:29–40, 2007.

[6] Akamai. The State of the Internet. White Paper, 2010. URL www.akamai.com. Last access on November 16, 2010.

[7] Nazareno Andrade, Miranda Mowbray, Aliandro Lima, Gustavo Wagner, and Matei Ripeanu. Influences on Cooperation in BitTorrent Communities. In *ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2005.

[8] Mauro Andreolini, Ricardo Lancellotti, and Philip S. Yu. Analysis of Peer-to-Peer Systems: Workload Characterization and Effects on Traffic Cacheability. In *12th IEEE Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS)*, 2004.

[9] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

[10] Siddhartha Annapureddy, Saikat Guha, Christos Gkantsidis, Dinan Gunawardena, and Pablo Rodriguez. Is High-quality VoD Feasible Using P2P Swarming? In *16th ACM International Conference on World Wide Web (WWW)*, 2007.

[11] Christina Aperjis, Michael J. Freedman, and Ramesh Johari. Peer-Assisted Content Distribution with Prices. In *4th ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2008.

[12] Luiz André Barroso and Urs Hölzle. The Case for Energy-proportional Computing. *IEEE Computer*, 40(12):33–37, 2007.

[13] Salman A. Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. *25th IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[14] Naimul Basher, Aniket Mahanti, Anirban Mahanti, Carey Williamson, and Martin Arlitt. A Comparative Analysis of Web and Peer-to-Peer Traffic. In *17th ACM International Conference on World Wide Web (WWW)*, 2008.

[15] Tim Berners-Lee. Net Neutrality: This is Serious. Timbl's Blog, 2006. URL http://dig.csail.mit.edu/breadcrumbs/blog/4. Last access on November 16, 2010.

[16] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. Analyzing and Improving a Bittorrent Network's Performance Mechanisms. In *25th IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[17] Ruchir Bindal, Pei Cao, and William Chan. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In *26th International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[18] Stevens Le Blond, Arnaud Legout, and Walid Dabbous. Pushing BitTorrent Locality to the Limit. *Computer Networks*, 55(3):541 – 557, 2011.

[19] Yacine Boufkhad, Fabien De Montgolfier, Fabien Mathieu, Orange Labs, Diego Perino, and Laurent Viennot. An Upload Bandwidth Threshold for Peer-to-Peer Video-on-Demand Scalability. In *23rd IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, 2009.

[20] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *18th IEEE Conference on Computer Communications (INFOCOM)*, 1999.

[21] Eric Bush, Max Schalcher, Peter Kühne, Stefan Kammermann, Stefan Gasser, and Jürg Nipkow. Measurement of the Power Consumption of Set-top Boxes. Swiss Federal Office of Energy, 2007. URL http//www.topten.ch/uploads/images/download-files/Schlussbericht-Settop-Boxen-V14_EN2-total.pdf. Last access on November 16, 2010.

[22] Niklas Carlsson, Derek L. Eager, and Anirban Mahanti. Peer-assisted On-demand Video Streaming with Selfish Peers. In *8th IFIP International Conferences on Networking (NETWORKING)*, 2009.

[23] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: High-bandwidth Multicast in Cooperative Environments. In *19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[24] Meeyoung Cha, Pablo Rodriguez, Sue Moon, and Jon Crowcroft. On Next-Generation Telco-Managed P2P TV Architectures. In *7th International Workshop on Peer-To-Peer Systems (IPTPS)*, 2008.

[25] Hyunseok Chang, Sugih Jamin, and Wenjie Wang. Live Streaming Performance of the Zattoo Network. In *9th ACM Internet Measurement Conference (IMC)*, 2009.

[26] Yih-Farn Chen, Yennun Huang, Rittwik Jana, Hongbo Jiang, Michael Rabinovich, Jeremy Rahe, Bin Wei, and Zhen Xiao. Towards Capacity and Profit Optimization of Video-on-Demand Services in a Peer-assisted IPTV Platform. *Multimedia Systems*, 15(1):19–32, 2008.

[27] Yih-Farn Robin Chen, Rittwik Jana, Daniel Stern, Bin Wei, Mike Yang, and Hailong Sun. Zebroid: Using IPTV Data to Support Peer-assisted VoD Content Delivery. In *19th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2009.

[28] Zhengjun Chena, Kaiping Xue, Peilin Hong, and Hancheng Lu. Differentiated Bandwidth Allocation for Reducing Server Load in P2P VOD. In *8th International Conference on Grid and Cooperative Computing*, 2009.

[29] Bin Cheng, Lex Stein, Hai Jin, and Zheng Zhang. Towards Cinematic Internet Video-on-Demand. In *3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008 (EuroSys)*, 2008.

[30] Huicheng Chi, Qian Zhang, Juncheng Jia, and Xuemin Shen. Efficient Search and Scheduling in P2P-based Media-on-Demand Streaming Service. *IEEE Journal on Selected Areas in Communications*, 25(1):119–130, 2007.

[31] Kenjiro Cho, Kensuke Fukuda, Hiroshi Esaki, and Akira Kato. Observing Slow Crustal Movement in Residential User Traffic. In *4th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2008.

[32] Yung Ryn Choe, Derek L. Schuff, Jagadeesh M. Dyaberi, and Vijay S. Pai. Improving VoD Server Efficiency with Bittorrent. In *ACM International Multimedia Conference (MM)*, 2007.

[33] David Choffnes and Fabian E. Bustamante. Taming the Torrent – A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[34] Cisco. Managing Peer-to-Peer Traffic with Cisco Service Control Technology. White Paper, 2005. URL http://www.cisco.com/en/US/prod/collateral/ps7045/ps6129/ps6133/ps6150/prod_white_paper0900aecd8023500d.pdf. Last access on November 16, 2010.

[35] Cisco. Visual Networking Index: Forecast and Methodology, 2009-2014. White Paper, 2010. URL http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf. Last access on November 16, 2010.

[36] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[37] Chris Dana, Danjue Li, David Harrison, and Chen-Nee Chuah. BASS: BitTorrent Assisted Streaming System for Video-on-Demand. In *7th IEEE Workshop on Multimedia Signal Processing (MMSP)*, 2005.

[38] Shirshanka Das, Saurabh Tewari, and Leonard Kleinrock. The Case for Servers in a Peer-to-Peer World. In *IEEE International Conference on Communications (ICC)*, 2006.

[39] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing Residential Broadband Networks. In *7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2007.

[40] André Dufour and Ljiljana Trajković. Improving Gnutella Network Performance using Synthetic Coordinates. In *3rd ACM International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)*, 2006.

[41] Jagadeesh M. Dyaberi, Karthik Kannan, and Vijay S. Pai. Storage Optimization for a Peer-to-Peer Video-on-Demand Network. In *1st ACM SIGMM Conference on Multimedia systems (MMSys)*, 2010.

[42] ENERGY STAR. Set-top Box Qualified Product List, 2010. URL http://www.energystar.gov/ia/products/prod_lists/set_top_boxes_prod_list.pdf. Last access on November 16, 2010.

[43] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Profiling a Million User DHT. In *7th ACM SIGCOMM Conference on Internet measurement (IMC)*, 2007.

[44] Gerhard Fettweis and Ernesto Zimmermann. ICT Energy Consumption – Trends and Challenges. In *11th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2010.

[45] Markus Fiedler, Tobias Hoßfeld, and Phuoc Tran-Gia. A Generic Quantitative Relationship between Quality of Experience and Quality of Service. *IEEE Network Special Issue on Improving QoE for Network Services*, 24(2):36 – 41, 2010.

[46] G-Lab. National Platform for Future Internet Studies, 2008. URL http://www.german-lab.de/. Last access on November 16, 2010.

[47] Prasanna Ganesan and Mukund Seshadri. On Cooperative Content Distribution and the Price of Barter. In *25th International Conference on Distributed Computing Systems (ICDCS)*, 2005.

[48] Pawel Garbacki, Dick H.J. Epema, Johan Pouwelse, and Maarten van Steen. Offloading Servers with Collaborative Video on Demand. In *7th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2008.

[49] Vijay Gopalakrishnan, Bobby Bhattacharjee, K.K. Ramakrishnan, Rittwik Jana, and Divesh Srivastava. CPM: Adaptive Video-on-Demand with Cooperative Peer Assists and Multicast. In *28th IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[50] Greg Goth. ISP Traffic Management: Will Innovation or Regulation Ensure Fairness? *IEEE Distributed Systems Online*, 9(9):1–4, 2008.

[51] Carsten Griwodz. *Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure*. PhD thesis, Technische Universität Darmstadt, 2000.

[52] Carsten Griwodz, Frank T. Johnsen, Simen Rekkedal, and Pål Halvorsen. Caching of Interactive Multiple Choice MPEG-4 Presentations. In *25th IEEE International Performance Computing and Communications Conference (IPCCC)*, 2006.

[53] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, and Xiaodong Zhang. The Stretched Exponential Distribution of Internet Media Access Patterns. In *27th ACM Symposium on Principles of Distributed Computing (PODC)*, 2008.

[54] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2Cast: Peer-to-Peer Patching Scheme for VoD Service. In *12th ACM International Conference on World Wide Web (WWW)*, 2003.

[55] Robert W. Hahn and Scott Wallsten. The Economics of Net Neutrality. *The Economists' Voice*, 3(6):1–7, 2006.

[56] Yensy James Hall, Patrick Piemonte, and Matt Weyant. Joost: A Measurement Study. Technical report, School of Computer Science, Carnegie Mellon University, 2007.

[57] Prashanth Hande, Mung Chiang, A. Robert Calderbank, and Sundeep Rangan. Network Pricing and Rate Allocation with Content Provider Participation. In *28th IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[58] David Harrison, Stanislav Shalunov, and Greg Hazel. BitTorrent Local Tracker Discovery Protocol, 2008. URL http://bittorrent.org/beps/bep_0022.html. Last access on November 16, 2010.

[59] Gerhard Hasslinger, Franz Hartleb, and Thomas Beckhaus. User Access to Popular Data on the Internet and Approaches for IP Traffic Flow Optimization. In *16th International Conference on Analytical and Stochastic Modeling Techniques and Applications, (ASMTA)*, 2009.

[60] Jiayue He, Augustin Chaintreau, and Christophe Diot. A Performance Evaluation of Scalable Live Video Streaming with Nano Data Centers. *Computer Networks*, 53(2):153–167, 2009.

[61] Oliver Heckmann. *The Competitive Internet Service Provider*. Wiley & Sons, 2006.

[62] Oliver Heckmann, Jens Schmitt, and Ralf Steinmetz. Optimizing Interconnection Policies. *Computer Networks*, 46(1):19–39, 2004.

[63] Noah Horowitz. NRDC Study of Set Top Box and Game Console Power Use, 2007. URL http://www.energystar.gov/ia/partners/prod_development/revisions/downloads/settop_boxes/NRDC_SetTopBox_Data_IEA.pdf. Last access on November 16, 2010.

[64] Tobias Hossfeld, Phuoc Tran-Gia, and Markus Fiedler. Quantification of Quality of Experience for Edge-Based Applications. In *20th International Teletraffic Congress (ITC)*, volume 4516, pages 361–373, 2007.

[65] Cheng Huang, Jin Li, and Keith W. Ross. Can Internet Video-on-Demand be Profitable? In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007.

[66] Cheng Huang, Jin Li, and Keith W. Ross. Peer-Assisted VoD: Making Internet Video Distribution Cheap. In *6th International Peer to Peer Symposium (IPTPS)*, 2007.

[67] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Understanding Hybrid CDN-P2P: Why Limelight Needs its Own Red Swoosh. In *18th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2008.

[68] Yan Huang, Tom Z.J. Fu, Dah-Ming Chiu, C. Huang, and John C.S. Lui. Challenges, Design and Analysis of a Large-scale P2P-VoD System. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[69] Intel. The All New 2010 Intel Core vPro Processor Family: Intelligence that Adapts to Your Needs. White Paper, 2010. URL http://download.intel.com/products/vpro/whitepaper/crossclient.pdf. Last access on November 16, 2010.

[70] International Telecommunication Union. Definition of Next Generation Network, 2004. URL http://www.itu.int/ITU-T/studygroups/com13/ngn2004/working_definition.html. Last access on November 16, 2010.

[71] ITU-T. H.264 : Advanced Video Coding for Generic Audiovisual Services, 2008. URL http://www.itu.int/rec/T-REC-H.264/. Last access on November 16, 2010.

[72] Raj K. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.

[73] Shakir James and Patrick Crowley. IMP: ISP-Managed P2P. In *10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2010.

[74] Vaishnav Janardhan and Henning Schulzrinne. Peer Assisted VoD for Set-top Box based IP Network. In *Peer-to-Peer Streaming and IP-TV Workshop (P2P-TV)*, 2007.

[75] Raul Jimenez, Lars-Erik Eriksson, and Björn Knutsson. P2P-Next: Technical and Legal Challenges, 2009. URL http://www.tslab.ssvl.kth.se/files/sncnw_main.pdf. Last access on November 16, 2010.

[76] Frank T. Johnsen, Trude Hafsoe, Carsten Griwodz, and Pal Halvorsen. Workload Characterization for News-on-Demand Streaming Services. In *26th IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2007.

[77] Guillaume Jourjon, Thierry Rakotoarivelo, and Max Ott. Models for an Energy-Efficient P2P Delivery Service. In *18th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2010.

[78] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2005.

[79] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki. Should Internet Service Providers Fear Peer-Assisted Content Distribution? In *5th ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2005.

[80] Sebastian Kaune, Tobias Lauinger, Aleksandra Kovacevic, and Konstantin Pussep. Embracing the Peer Next Door: Proximity in Kademlia. In *8th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2008.

[81] Sebastian Kaune, Konstantin Pussep, Gareth Tyson, Andreas Mauthe, and Ralf Steinmetz. Cooperation in P2P Systems through Sociological Incentive Patterns. In *3rd International Workshop on Self-Organizing Systems (IWSOS)*, 2008.

[82] Sebastian Kaune, Gareth Tyson, Konstantin Pussep, Andreas Mauthe, and Ralf Steinmetz. The Seeder Promotion Problem: Measurements, Analysis and Solution Space. In *19th International Conference on Computer Communications and Networks (ICCCN)*, 2010.

[83] Ram Keralapura, Nina Taft, and Gianluca Iannaccone Chen-Nee Chuah. Can ISPs Take the Heat from Overlay Networks? In *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[84] Ken Kerpez, Yuanqiu Luo, and Frank J. Effenberger. Bandwidth Reduction via Localized Peer-to-Peer (P2P) Video. *International Journal of Digital Multimedia Broadcasting*, 2010:1–11, 2010.

[85] Kontiki. Power of Commercial Peer-to-Peer Delivery. White Paper. URL www.kontiki.com. Last access on November 16, 2010.

[86] Dejan Kostic, Adolfo Rodriguez, Jeannie R. Albrecht, and Amin Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[87] Aleksandra Kovacevic, Oliver Heckmann, Nicolas Liebau, and Ralf Steinmetz. Location Awareness - Improving Distributed Multimedia Communication. *Special Issue of the Proceedings of IEEE on Advances in Distributed Multimedia Communications*, 96(1), 2008.

[88] Rakesh Kumar, Yong Liu, and Keith W. Ross. Stochastic fluid theory for P2P streaming systems. In *26th IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[89] Craig Labovitz, S. Iekel-Johnson, D. McPherson, Jon Oberheide, and Farnam Jahanian. Internet Inter-domain Traffic. *ACM SIGCOMM Computer Communication Review*, 40(4):75–86, 2010.

[90] Nikolaos Laoutaris, Pablo Rodriguez, and Laurent Massoulié. ECHOS: Edge Capacity Hosting Overlays of Nano Data Centers. *ACM SIGCOMM Computer Communication Review*, 38:51–54, 2008.

[91] Uichin Lee, Ivica Rimac, and Volker Hilt. Greening the Internet with Content-Centric Networking. In *1st International Conference on Energy-Efficient Computing and Networking (e-Energy)*, 2010.

[92] Frank Lehrieder, Tobias Hossfeld, Simon Oechsner, and Vlad Singeorzan. The Impact of Caching on BitTorrent-Like Peer-to-Peer Systems. In *10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2010.

[93] Frank Lehrieder, Simon Oechsner, Tobias Hoßfeld, Zoran Despotovic, Wolfgang Kellerer, and Maximilian Michel. Can P2P-Users Benefit from Locality-Awareness? In *10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2010.

[94] Jun Lei, Lei Shi, and Xiaoming Fu. An Experimental Analysis of Joost Peer-to-Peer VoD Service. *Peer-to-Peer Networking and Applications*, 3(4):351–362, 2010.

[95] Nathaniel Leibowitz, Aviv Bergman, Roy Ben-Shaul, and Aviv Shavit. Are File Swapping Networks Cacheable? Characterizing P2P Traffic. In *7th International Workshop on Web Content Caching and Distribution (WCW)*, 2002.

[96] Dave Levin, Katrina LaCurts, Neil Spring, and Bobby Bhattacharjee. BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[97] Bo Li, Susu Xie, Yang Qu, Gabriel Yik Keung, Chuang Lin, Jiangchuan Liu, and Xinyan Zhang. Inside the New Coolstreaming: Principles, Measurements and Performance Implications. In *27th IEEE Conference on Computer Communications (INFOCOM)*, 2008.

[98] Chao Liang, Zhenghua Fu, and Yong Liu. iPASS : Incentivized Peer-assisted System for Asynchronous Streaming. In *28th IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[99] Yong Liu, Yang Guo, and Chao Liang. A Survey on Peer-to-Peer Video Streaming Systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, 2008.

[100] Macworld.com. Wake on Demand lets Snow Leopard sleep with one eye open, 2009. URL http://www.macworld.com/article/142468/2009/08/wake_on_demand.html. Last access on November 16, 2010.

[101] MaxMind. GeoIP Databases. URL http://www.maxmind.com/. Last access on November 16, 2010.

[102] Alan Meier. Standby Power. URL http://standby.lbl.gov. Last access on November 16, 2010.

[103] David Meisner, Brian T. Gold, and Thomas F. Wenisch. PowerNap: Eliminating Server Idle Power. In *14th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2009.

[104] Klaus Mochalski and Hendrik Schulze. Deep Packet Inspection – Technology, Applications & Net Neutrality. ipoque GmbH: White Paper, 2009. URL http://www.ipoque.com/userfiles/file/DPI-Whitepaper.pdf. Last access on November 16, 2010.

[105] Jan David Mol, Johan Pouwelse, Michel Meulpolder, Dick Epema, and Henk Sips. Give-to-Get: An Algorithm for P2P Video-on-Demand. In *15th Multimedia Computing and Networking (MMCN)*, 2008.

[106] Tatsuya Mori, Noriaki Kamiyama, and Shigeaki Harada. Improving Deployability of Peer-assisted CDN Platform with Incentive. In *IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.

[107] Matthew Moskovciak and David Katzmaier. PS3 Slim Uses Half the Power of PS3 Fat, 2009. URL http://news.cnet.com/8301-17938_105-10318727-1.html. Last access on November 16, 2010.

[108] San Murugesan. Harnessing Green IT: Principles and Practices. *IT professional*, 10(1):24–33, 2008.

[109] Suresh K. Nair and David C. Novak. A Traffic Shaping Model for Optimizing Network Operations. *European Journal of Operational Research*, 180(3):1358–1380, 2007.

[110] Multichannel News. YouTube May Lose $470 Million In 2009, 2009. URL http://www.multichannel.com/article/191223-YouTube_May_Lose_470_Million_In_2009_Analysts.php. Last access on November 16, 2010.

[111] Di Niu, Baochun Li, and Shuqiao Zhao. Self-Diagnostic Peer-Assisted Video Streaming through a Learning Framework. In *ACM International Conference on Multimedia (ACM Multimedia)*, pages 73–82, 2010.

[112] William B. Norton. A Business Case for ISP Peering. White Paper (v1.3), 2002. URL http://drpeering.net/white-papers/_pdfs/A-Business-Case-for-Peering.pdf. Last access on November 16, 2010.

[113] Andrew Odlyzko. Network Neutrality, Search Neutrality, and the Never-ending Conflict between Efficiency and Fairness in Markets. *Review of Network Economics*, 8(1):40–60, 2009.

[114] Simon Oechsner, Frank Lehrieder, Tobias Hossfeld, Florian Metzger, Konstantin Pussep, and Dirk Staehle. Pushing the Performance of Biased Neighbor Selection through Biased Unchoking. In *9th International Conference on Peer-to-Peer Computing (P2P)*, 2009.

[115] Ookla. Net Index, 2010. URL http://www.netindex.com/. Last access on November 16, 2010.

[116] Oversi. P2P Content Delivery Solutions. White Paper, 2007. URL http://www.oversi.com/images/stories/white_paper_july.pdf. Last access on November 16, 2010.

[117] John M. Peha. The Benefits and Risks of Mandating Network Neutrality, and the Quest for a Balanced Policy. *International Journal of Communication*, 1:644–668, 2007.

[118] Ryan S. Peterson and Emin Guün Sirer. Antfarm: Efficient Content Distribution with Managed Swarms. In *6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[119] Michael Piatek and Richard Yang. Contracts: Practical Contribution Incentives for P2P Live Streaming. In *7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010.

[120] Michael Piatek, Harsha V. Madhyastha, John .P. John, Arvind Krishnamurthy, and Thomas Anderson. Pitfalls for ISP-friendly P2P design. In *8th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2009.

[121] Thomas Plagemann, Vera Goebel, Carsten Griwodz, Pal Halverson, Laurent Mathy, Nicholas Race, and Michael Zink. Towards Scalable and Affordable Content Distribution Services. In *7th IEEE International Conference on Telecommunications (ConTEL)*, 2003.

[122] Jack Prins. e-Handbook of Statistical Methods. In Carroll Croarkin and Paul Tobias, editors, *6.3.2.4. EWMA Control Charts*. NIST/SEMATECH, 2010. URL http://www.itl.nist.gov/div898/handbook/pmc/section3/pmc324.htm. Last access on November 16, 2010.

[123] Konstantin Pussep, Simon Oechsner, Osama Abboud, Miroslaw Kantor, and Burkhard Stiller. Impact of Self-Organization in Peer-to-Peer Overlays on Underlay Utilization. In *4th International Conference on Internet and Web Applications and Services (ICIW)*, 2009.

[124] Peter Racz, Simon Oechsner, and Frank Lehrieder. BGP-Based Locality Promotion for P2P Applications. In *19th IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2010.

[125] Joshua Reich, Michel Goraczko, Aman Kansal, and Jitendra Padhye. Sleepless in Seattle No Longer. *USENIX Annual Technical Conference (USENIX ATC)*, 2010.

[126] Christian Reimsbach-Kounatze. Towards Green ICT Strategies: Assessing Policies and Programmes on ICTs and the Environment. *OECD Digital Economy Papers*, 2009. URL http://ideas.repec.org/p/oec/stiaab/155-en.html. Last access on November 16, 2010.

[127] Don Reisinger. Daily Tidbits: Joost Kills Software Application, 2008. URL http://news.cnet.com/8301-17939_109-10125447-2.html#ixzz14DSuFOxD. Last access on November 16, 2010.

[128] Ivica Rimac, Anwar Elwalid, and Sem Borst. On Server Dimensioning for Hybrid P2P Content Distribution Networks. In *8th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2008.

[129] Pablo Rodriguez, See-Mong Tan, and Christos Gkantsidis. On the Feasibility of Commercial, Legal P2P Content Distribution. *Computer Communication Review*, 36(1):75–78, 2005.

[130] RTSECURITY.COM. Online Wake-On-LAN. URL http://www.wakeonlan.me/. Last access on November 16, 2010.

[131] Osama Saleh and Mohamed Hefeeda. Modeling and Caching of Peer-to-Peer Traffic. In *14th IEEE International Conference on Network Protocols (ICNP)*, 2006.

[132] Hendrik Schulze and Klaus Mochalski. Internet Study 2008/2009. ipoque GmbH, 2009. URL http://www.ipoque.com/study/?t=ipqstudy. Last access on November 16, 2010.

[133] Jan Seedorf and Eric Burger. Application-Layer Traffic Optimization (ALTO) Problem Statement. IETF RFC 5693, 2009. URL https://datatracker.ietf.org/doc/draft-ietf-alto-problem-statement/. Last access on November 16, 2010.

[134] Jan Seedorf, Sebastian Kiesel, and Martin Stiemerling. Traffic Localization for P2P-applications: The ALTO Approach. In *9th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2009.

[135] Jan Seedorf, Saverio Niccolini, Martin Stiemerling, Ettore Ferranti, and Rolf Winter. Quantifying Operational Cost-Savings through ALTO-Guidance for P2P Live Streaming. In *3rd Workshop on Economic Traffic Management (ETM)*, 2010.

[136] SmoothIT. Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies, 2008. URL http://www.smoothit.org. Last access on November 16, 2010.

[137] Robin Sommer and Anja Feldmann. NetFlow: Information Loss or Win? In *2nd ACM SIGCOMM Workshop on Internet Measurement (IMW)*, 2002.

[138] Sergios Soursos, Maria Angeles Callejo Rodriguez, Konstantin Pussep, Peter Racz, Spiros Spirou, George D. Stamoulis, and Burkhard Stiller. ETMS: A System for Economic Management of Overlay Traffic. In *Towards the Future Internet - Emerging Trends from European Research*. IOS press, 2010.

[139] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications*. Springer, 2004.

[140] Mark Stuart. Challenges and Opportunities of Internet Television: Optimizing Peer-to-Peer Technologies for Set-Top Boxes. In *10th IEEE International Conference on P2P Computing (P2P)*, 2010.

[141] Ao-Jan Su, David R. Choffnes, Aleksandar Kuzmanovic, and Fabian E. Bustamante. Drafting behind Akamai (Travelocity-based Detouring). In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2006.

[142] Kyoungwon Suh, Christophe Diot, Jim Kurose, Laurent Massoulié, Christoph Neumann, Donald F. Towsley, and Matteo Varvello. Push-to-Peer Video-on-Demand System: Design and Evaluation. *IEEE Journal on Selected Areas in Communications*, 25(9):1706–1716, 2007.

[143] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, fourth edition, 2002.

[144] Saurabh Tewari and Leonard Kleinrock. Proportional Replication in Peer-to-Peer Networks. In *25th IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[145] TorrentFreak. Comcast Throttles BitTorrent Traffic, Seeding Impossible, 2007. URL http://torrentfreak.com/comcast-throttles-bittorrent-traffic-seeding-impossible/.

[146] Phuoc Tran-Gia, Towards Hossfeld, Michael Menth, and Rastin Pries. Emerging Issues in Current Future Internet Design. *e&i Elektrotechnik und Informationstechnik*, 126(7):241–249, 2009.

[147] Tribler Client, 2010. URL http://tribler.org/. Last access on November 16, 2010.

[148] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulie, Christophe Diot, and Pablo Rodriguez. Greening the Internet with Nano Data Centers. In *5th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2009.

[149] Aggelos Vlavianos, Marios Iliofotou, and Michalis Faloutsos. BiToS: Enhancing BitTorrent for supporting Streaming Applications. In *25th IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[150] Jiajun Wang, Cheng Huang, and Jin Li. On ISP-friendly Rate Allocation for Peer-assisted VoD. In *16th ACM International Conference on Multimedia (ACM Multimedia)*, 2008.

[151] Martin B. Weiss and Seung Jae Shin. Internet Interconnection Economic Model and its Analysis: Peering and Settlement. *NETNOMICS: Economic Research and Electronic Networking*, 6(1):43–57, 2004.

[152] Azureus Wiki. Bad ISPs. URL http://wiki.vuze.com/w/Bad_ISPs. Last access on November 16, 2010.

[153] Dave Winer. XML-RPC Specification, 1999. URL http://www.xmlrpc.com/spec. Last access on November 16, 2010.

[154] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2005.

[155] Chuan Wu, Baochun Li, and Shuqiao Zhao. Multi-channel live p2p streaming: refocusing on servers. In *27th IEEE Conference on Computer Communications (INFOCOM)*, 2008.

[156] Jiahua Wu and Baochun Li. Keep Cache Replacement Simple in Peer-Assisted VoD Systems. In *28th IEEE Conference on Computer Communications (INFOCOM) (minitrack)*, 2009.

[157] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[158] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky. In *ACM International Conference on Multimedia (ACM Multimedia)*, 2009.

[159] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. Understanding User Behavior in Large-scale Video-on-Demand Systems. In *1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys)*, 2006.

[160] Hao Zhang and Kannan Ramchandran. A Reliable Decentralized Peer-to-Peer Video-on-Demand System Using Helpers. *IEEE Picture Coding Symposium (PCS)*, 2009.

[161] Hubert Zimmermann. OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communication*, 28(4): 425–432, 1980.

[162] Michael Zink. *Scalable Internet Video-on-Demand Systems*. PhD thesis, Technische Universität Darmstadt, 2003.

[163] Michael Zink, Kyoungwon Suh, Yu Gu, and Jim Kurose. Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications. *Computer Networks*, 53(4):501–514, 2009.

[164] Thomas Zinner, Simon Oechsner, Tobias Hossfeld, and Phuoc Tran-Gia. On the Trade-Off between Efficiency and Congestion in Location-Aware Overlay Networks-Example of a Vertical Handover Support System. In *8th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2008.

[165] Thomas Zinner, Osama Abboud, Oliver Hohlfeld, Tobias Hossfeld, and Phuoc Tran-Gia. Towards QoE Management for Scalable Video Streaming. In *21th ITC Specialist Seminar on Multimedia Applications-Traffic, Performance and QoE*, 2010.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ALTO	Application-Layer Traffic Optimization
AS	Autonomous System
BGP	Border Gateway Protocol
CDN	Content Delivery Network
DPI	Deep Packet Inspection
DSLAM	Digital Subscriber Line Access Multiplexer
DSL	Digital Subscriber Line
DVR	Digital Video Recorder
FIFO	First-In First-Out
G2G	Give-to-Get
Gbps	Gigabit per second
GB	GigaByte
HAP	Highly Active Peer
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IPTV	Internet Protocol TeleVision
IP	Internet Protocol
ISP	Internet Service Provider
IXP	Internet eXchange Point
kbps	kilobit per Second
LAN	Local Area Network
LFU	Least Frequently Used
LOAPS	LOcality-Aware Peer Selection
LRU	Least Recently Used
MAC	Media Access Control
Mbps	Megabit per Second
MWh	MegaWatt hour
NGN	Next Generation Networking
p2p	peer-to-peer
P4P	Proactive network Provider Participation for P2p
PC	Personal Computer

QoE	Quality of Experience
QoS	Quality of Service
SARU	Supply-Aware Recently Used
SLARU	Supply- and Locality-Aware Recently Used
STB	Set-top Box
TB	Terrabyte
VoD	Video-on-Demand
VoIP	Voice over IP

APPENDIX

A.1 FURTHER EVALUATION RESULTS FOR CHAPTER 4

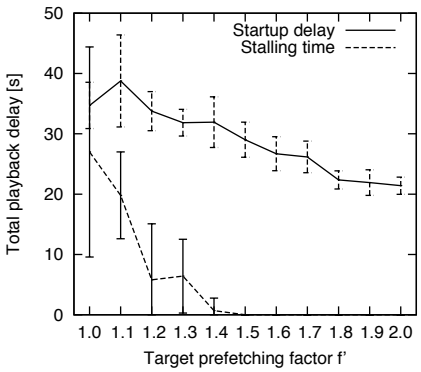
This section presents supplementary experimental results for Chapter 4. They include (1) performance study of the global speed policy, (2) performance study of the supporter policy, and (3) comparison of adaptive and static allocation policies in a flash-crowd scenario.

A.1.1 *Global Speed Policy*

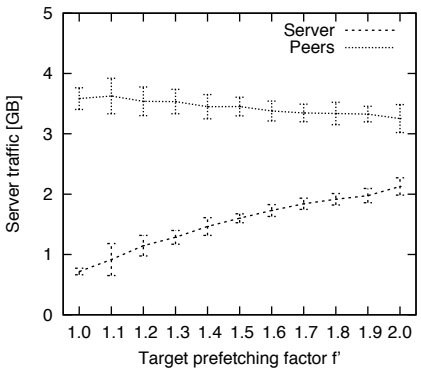
In this subsection, we analyze the performance of the global speed allocation policy with respect to the desired *target* speed. As already explained in Section 4.4.1 this policy tries to keep the global speed close to the desired target where the target is defined as f' times video bitrate. If f' is set too low, some nodes might experience undesired playback delays. If f' is chosen too high, too many servers will join the swarm. We expect to find a suitable value in the range $[1:2]$, where 1 corresponds to no capacity for prefetching and 2 allows fast prefetching.

In order to find suitable values of f' in the utilized VoD system, we ran a series of experiments with varying parameter values. Figure 46 shows the 95th percentiles of startup delay and stalling time (see Figure 46a) and the relative upload contribution of the servers (Figure 46b) for different values of f'.

We observe that f' values close to 1 result in high playback delays and unstable performance (high standard deviation), while keeping the server's contribution low. Peers, whose average upload rate is close to the playback rate, provide most of the resources but cannot keep up with the arrival and departure rates. We further observe that with the increasing f' the server traffic increases faster than the peer load decreases. This stems from the too high server resources allowing peers to prefetch content fast. Since most peers leave after watching roughly half of the video, an aggressive prefetching possibility leads to a lot of segments being prefetched but never watched.

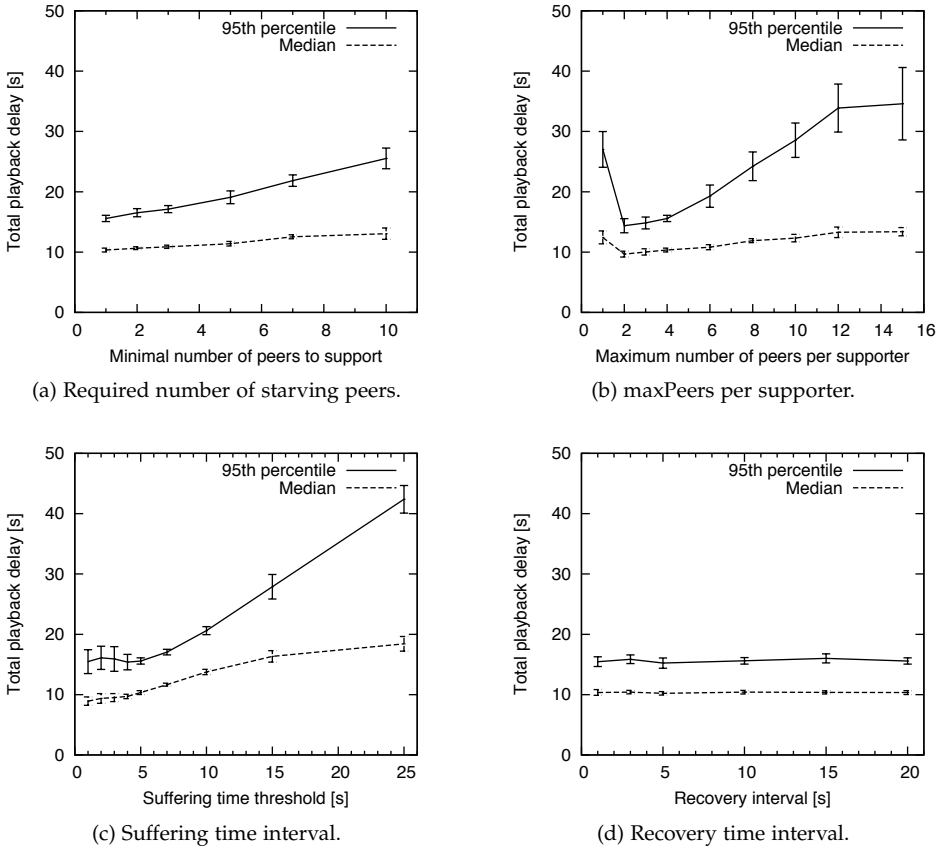


(a) Stalling times and startup delays (95th percentiles).



(b) Server traffic.

Figure 46: Impact of target speed factor on playback delay and upload.



(a) Required number of starving peers.

(b) maxPeers per supporter.

(c) Suffering time interval.

(d) Recovery time interval.

Figure 47: Impact of various parameters on the supporter policy performance.

Nevertheless, high f' results in much lower playback delays, while the server traffic grows significantly. We further see that f' equal to 1.5 is sufficient to avoid stalling, and achieve short startup delays, while avoiding unnecessary load at servers. These observations justify the usage of $f' = 1.5$ as operating point of this policy.

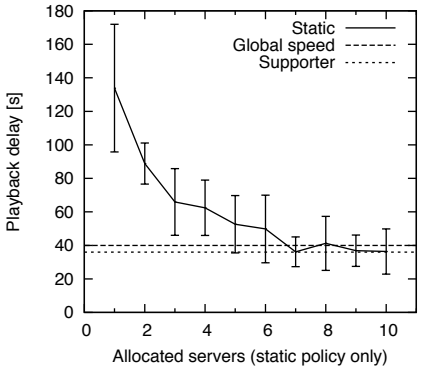
A.1.2 *Supporter Policy*

For the supporter policy we are interested in understanding the impact of the following parameters:

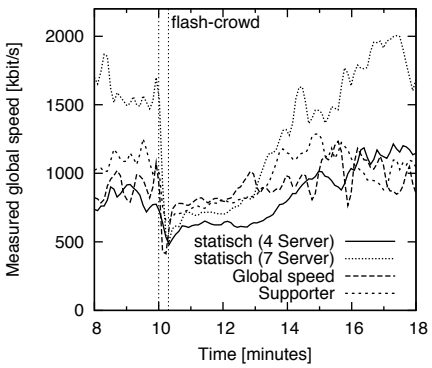
1. *minStarving*: How many suffering peers must be present to allocate an additional server,
2. *maxPeers*: Maximum number of peers a supporter can take care of,
3. *sufferTime*: Time interval with not full playout buffer to consider a peer as suffering,
4. *recoveryTime*: Time interval with filled buffer after which a previously suffering peer is considered as recovered.

To evaluate the impact of these parameters, we fix the default configuration as follows: *minStarving* = 1, *maxPeers* = 4, *sufferTime* = 5 seconds, and *recoveryTime* = 20 seconds. Then we subsequently modify single values as demonstrated in Figure 47.

Figure 47a shows the impact of the minimum number of suffering peers to allocate an additional server. We observe that the median is quite insensitive with regard to



(a) Total playback delay (95th percentiles).



(b) Server allocation over time.

Figure 48: Impact of flash crowds (after 10 minutes 80 additional peers arrive within 20 seconds).

this parameter, while the values around 2 and 3 peers prevent too bad performance for outliers. This can be explained by the fact, that waiting for too many suffering peers results in bad performance for single peers in the suffering state.

The impact of the maximum number of peers to be handled by one supporter is presented in Figure 47b. We can observe that the best delays are achieved if the supporter handles only 2-4 peers. Beyond 6 peers per supporter the 95th percentiles increases dramatically. At the first spot the performance increase when going from 1 to 2 peers per supporter might appear counterintuitive. However, allocating the whole supporter capacity to a single peer results in too fast prefetching of segments and, therefore, wasted upload capacity, since a peer might depart without watching and uploading them to other peers.

Another interesting point is how long a peer should try to fill its buffer, until being considered as *suffering* by the indexing server. It turns out that values around 4 seconds are optimal (see Figure 47c). This is roughly the half of the playout buffer size and, therefore, corresponds to our expectation that supporter should react on the suffering state before playback stalls occur.

Finally, Figure 47d shows the recovery period with values between 1 and 20 seconds. However, this parameter does not exhibit a significant impact in our workload. Both the median and the 95th percentiles are very close among the setup.

A.1.3 *Impact of Flash-Crowds*

Flash-crowd effects describe a steep increase in popularity of certain videos, which can happen, for instance, with news trailers or user-generated content published on popular blog sites [97, 37, 10]. Such effects put significant stress on the streaming system and are therefore important to consider for any adaptation algorithm that should work in a scalable and robust manner.

To understand the impact of flash-crowds on the server allocation, we use the workload where the arrival rate of peers increases by 2000% after 10 minutes of the simulation run. This increased load lasts only 20 seconds resulting in 80 additional peers. We reuse the policy configurations from the previous experiment and again vary the number of statically allocated servers. The resulting playback delay is shown in Figure 48a. For the sake of clarity only the total delay is presented (more details for selected setups are presented in Table 16). The previously best static allocation with 4 servers results now in a much worse playback performance (total delay of 60 vs. 30.9 seconds as observed in Figure 15 with 4 static servers). In this workload, the best

Table 16: Performance comparison of various policies in the *flash-crowd workload*.

Policy	Server traffic [GB]	Reporting overhead [MB]	Total Delay [s]		Stalling Time [s]	
			50%	95%	50%	95%
Static (4 servers)	1.82	-	14.5	62.5	0	11.8
Static (7 servers)	2.68	-	8.5	36.2	0	1.7
Global speed	1.78	26.70	12.7	41.0	0	8.5
Supporter	2.01	29.93	11.4	36.7	0	0

static performance is achieved with seven statically allocated servers, which is our new benchmark to assess the performance of adaptive allocation in the considered flash-crowd scenario.

Figure 48a also demonstrate that the adaptive allocation policies result in a comparable streaming performance of 36 and 41 seconds, respectively (cf. 36 seconds of the best static policy). Figure 48b further shows the average download speed of the swarm during the simulation. The flash-crowd is clearly visible 10 minutes after the experiment start. We also observe that the adaptive policies achieve similar average download speeds as the less efficient static policy with seven servers, but (as shown in Table 16) the playback performance is close to the best static policy with 4 servers. This is the case both with regard to the total delay and stalling time. Again we observe that the supporter policy outperforms the global speed policy and is close to the best static allocation. Inspection of the measured standard deviations (not shown in in Table 16) revealed only roughly 10 seconds for 50th and only 1 second for the 95th percentiles for all policies.

Considering the costs of the policies we observe that adaptive policies require less segments uploaded from the servers than the best static policy (25% savings for the supporter and 34% savings for the global speed policy). We also measured the reporting overhead induced by our policies, which turned out to be only 1.5% of the content server traffic. Furthermore, the supporter policy introduces 27.5% larger indexing server overhead than the global speed policy (cf. Table 16, column *Reporting*). However, this overhead is outweighed by the improved performance and the fact that the content provider does not need to know the user demand in advance (in order to estimate the number of required static servers).

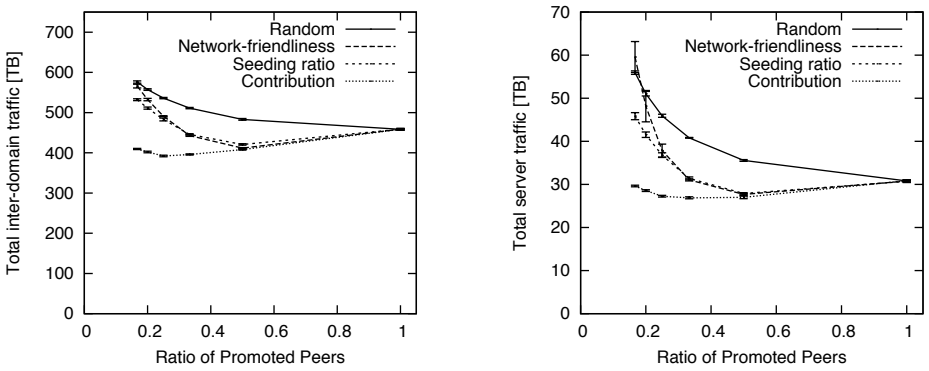
In summary, the flash-crowd workload stresses the streaming system significantly by increasing the median delay. Still, the supporter policy again offers good trade-off between server traffic and playback performance especially with regard to stalling time and outliers. The performance is better than that of the global speed policy and very close to that of the best static server allocation.

A.2 FURTHER EVALUATION RESULTS FOR CHAPTER 5

This section presents additional simulation results for Chapter 5. The experimental setup is the same as described in Section 5.6.1.

A.2.1 *Dimensioning of Highly Active Peers*

In this experiment we evaluate the trade-off between promotion ratio and promotion factor. For this purpose, we fix the promotion volume to 100% (i.e. double the total upload capacity of users) and modify the promotion ratio and promotion factor simultaneously according to Equation 5.2. The resulting *(ratio, factor)* tuples are: (100%,2x), (50%,3x), (33.3%,4x), (25%,5x), (20%,6x), and (16.6%,7x).



(a) Inter-domain traffic.

(b) Server traffic.

Figure 49: Impact of HAP selection metrics with varying promotion volume (promotion ratio fixed to 5).

Figure 49 shows the performance of basic HAP selection metric for each of the resulting configurations. Our main observations are:

1. With all considered configurations, the contribution metric outperforms other HAP selection metrics, while the network-friendliness and seeding ratio metrics are still better than random selection. This applies both to the inter-domain and server traffic.
2. The best performance for the network operators is achieved with the promotion ratio of 25% (and 5x promotion factor).
3. The best performance for the overlay is achieved with the promotion ratio of 33.3% (and 4x promotion factor).
4. If more than 25–33% of peers are promoted, the differences between the non-random metrics disappear.

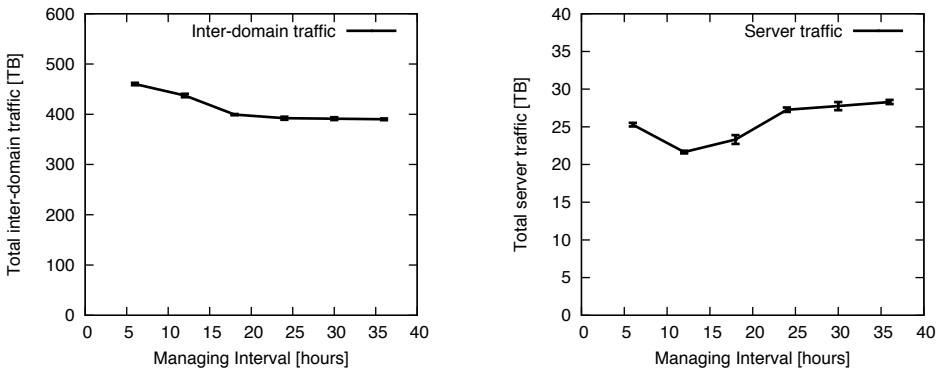
Recalling that heavy users constitute 20% of the overlay, we conclude that the promotion ratio should roughly correspond to the amount of such users in the system. Since their ratio can be expected to be quite stable over the time, a network operator can configure the promotion ratio based on mid-term measurements of user behavior.

A.2.2 *Management Interval*

This experiments analyzes a suitable value of HAP management interval. This covers both the measurement interval and the HAP promotion interval that determines when

peers are promoted and demoted. The interval is varied between 6 and 36 hours, while all other parameters are set to their default values. We recall that the management interval feasible from the technical point of view (to re-configure user access profiles for promoted and demoted peers) ranges in the area.

Figure 50 presents the results obtained. We observe that the inter-domain traffic decreases with increasing traffic management interval (see Figure 50a). However, from 18 hours upwards this decrease is only slight, so that our initial proposal to use 24 hours interval is a good choice. The server traffic decreases significantly up to an interval of 12 hours and then increases again (see Figure 50b). We attribute this behavior to the alignment between the traffic management interval and the idle and busy phases, which both have the length of 12 hours, too. that get promoted while they are actively taking part in the system can immediately make use of that increase and, thus, reduce server traffic.



(a) Inter-domain traffic.

(b) Server traffic.

Figure 50: Impact of HAP Management Interval.

Because network operators choose the promotion interval, we focus on the effects that the management interval has on inter-domain traffic. A short interval is costly and does not benefit the network operator as much as a longer interval. The reason is that a short interval makes promotion behaving unstable by demoting promoted peers before they could make use of the additional bandwidth. However, choosing the interval too long would result in a lack in dynamic adaptability to changing circumstances. Therefore, we believe that the network operator sees 24 hours as an appropriate value, which is also our default value, and allows, for example, changing user profiles during the idle morning hours without affecting running applications.

A.3 FURTHER EVALUATION RESULTS FOR CHAPTER 6

In this section we present further simulation results for an energy-aware utilization of STBs in a peer-assisted VoD in addition to the experiments discussed in Section 6.7. The basic setup follows again the methodology presented in Section 6.6.

A.3.1 *Scalability of Standby Policies*

In the first experiments we consider the impact of various external factors on the performance of standby policies. Each sub-experiment uses the same basic setup and varies only the considered parameter.

The percentage of data that must be served by content servers depends on the peers' upload capacity, especially in comparison with the video bitrate. If the upload bandwidth is higher than the video bitrate most of the data can be served by peers, while otherwise servers has to provide more bandwidth. To assess this trade-off we keep the video bitrate of the basic scenario (the average bitrate is 2 Mbps and vary the upload bandwidth of peers in the range of $[250 : 4000]$ kbps, which means 1/8 to 2 times the video bitrate[1]. This range allows for both the currently widespread slower and upcoming faster access networks. Furthermore, knowing the average upload bandwidth of its customers the content provider can dimension the bitrate of its videos to assure that the server traffic will stay in the acceptable range.

Figure 51 presents the impact of the available peer upload bandwidth and compares the energy savings for each policy relative to the always-on policy. Here, *the savings increase with the higher upload bandwidth*, which can be explained by *less online peers required to serve the same amount of requests*. We observe that all policies show a saturation effect for the bandwidth around 2 Mbps, since here each peer can upload one or two complete streams in parallel (video bitrate is in $\{1, 2, 4\}$ Mbps). Still, the SARU policy is able to save 6% more energy than the non-adaptive selfish and overtime policies, coming very close to the optimal wake-on-demand policy.

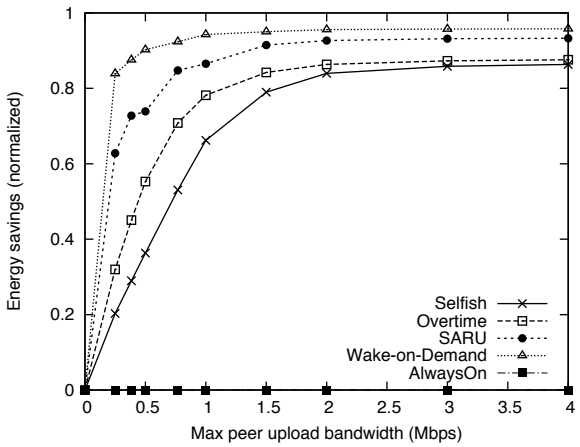


Figure 51: Energy savings compared to the always-on behavior (normalized) with varying upload bandwidth per peer.

For lower upload capacities the difference between the policies is much higher. For example, an upload bandwidth of 256 kbps per peer results in 50% savings with the SARU but only 18% with the selfish policy. In such a low-dimensioned scenario, streaming of a single video requires $4 - 12$ online STBs to cover the demand. We conclude that our heuristic can outperform the non-adaptive policies and offers the

1 We also include the results with upload bandwidth set to zero, in which case their is no effect of peer-assistance.

savings comparable to the wake-on-demand policy independent of the peer bandwidth.

In this experiment we analyze the scalability of standby policies by considering the impact of the population size. For this purpose, we generate random subsets of the international trace with different number of peers (between 5,000 to 60,000) and investigate the energy savings (see Figure 52).

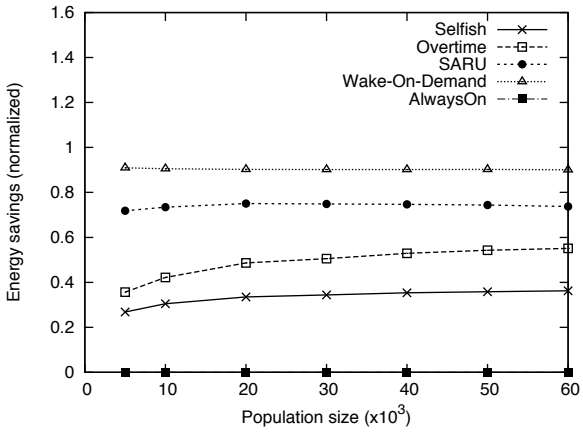


Figure 52: Energy savings compared to the always-on behavior (normalized) as a function of population size.

We observe that *the increasing population size leads to higher savings*. This can be explained by the fact that for our traces the number of peers grows faster than the number of requested videos (2,620 videos for 60,000 peers vs. 1,260 videos for 10,000 peers) resulting in more requests per video (7.9 requests per video with 10,000 peers but 22.9 requests per video with for 60,000 peers). This yields a higher replication degree, which allows more peers to go offline without hurting the overlay performance. Furthermore, while all policies are able to improve their performance, *the wake-on-demand policy is again the best one and SARU offers a good trade-off outperforming the overtime and selfish policies*.

A.3.2 *Performance with a National Workload*

This experiment compares the performance of standby policies for the national workload. The goal is to confirm the basic results with regard to the policies' comparative performance, presented in Section 6.7.1 for the international workload.

Indeed, while the absolute values differ slightly, the relative performance is very similar. Again, the wake-on-demand policy turns out to be the most efficient policy with regard to both the server traffic and normalized energy savings. The SARU policy performs similar to wake-on-demand, while the other policies suffer either from a higher server traffic or too long online time.

A.3.3 *Additional Results for Locality Awareness*

This section provides supplementary results for the experiment with locality-aware extension of the SARU standby policy (see Section 6.7.4). Figure 54 shows the relative online time and relative server traffic for SARU and SLARU either with or without locality-aware peer selection (LOAPS).

The main observations are that LOAPS has no effect on the relative online time, while SLARU extends the online time by factor two (for the national peers) and factor

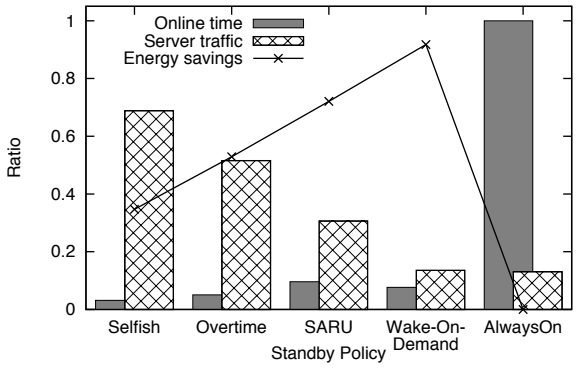
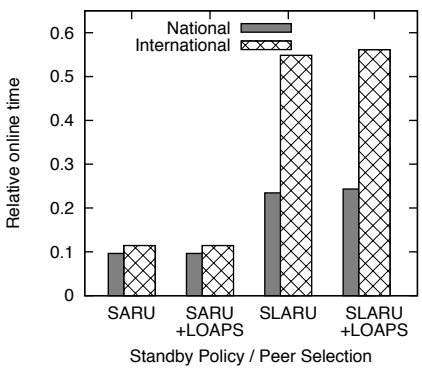
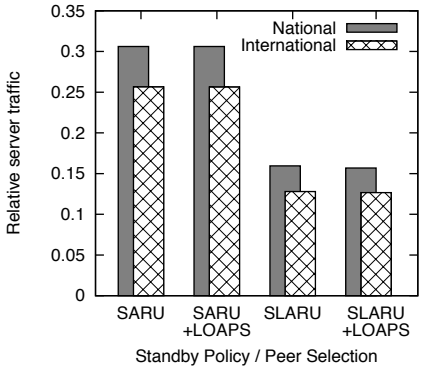


Figure 53: Relative online time and resulting server traffic of various policies (german peers only).



(a) Relative online time.



(b) Relative server traffic.

Figure 54: Impact of locality awareness on online time and server traffic with national and international peers (SLARU = locality-aware SARU, LOAPS = locality-aware peer selection).

five for international peers (see Figure 54a). The reason is again the smaller amount of peers per network domain with national peers. Furthermore, the server traffic is reduced in both cases roughly by factor two (see Figure 54b). In case of the national workload it outweighs the increased online time.

B AUTHOR'S PUBLICATIONS

B.1 MAIN PUBLICATIONS

[1] Konstantin Pussep, Osama Abboud, Florian Gerlach, Ralf Steinmetz, and Thorsten Strufe. Adaptive Server Allocation for Peer-assisted VoD. In *24th IEEE International Parallel and Distributed Processing Symposium, Workshops and Phd Forum (IPDPSW)*. 2010.

[2] Konstantin Pussep, Sebastian Kaune, Osama Abboud, Christian Huff, and Ralf Steinmetz. On Energy-Awareness for Peer-assisted Streaming with Set-Top Boxes. In *6th International Conference on Network and Service Management (CNSM)*. 2010.

[3] Konstantin Pussep, Sergey Kuleshov, Christian Groß, and Sergios Soursos. An Incentive-based Approach to Traffic Management for Peer-to-Peer Overlays. In *3rd Workshop on Economic Traffic Management (ETM)*. 2010.

[4] Konstantin Pussep, Christof Leng, and Sebastian Kaune. Modeling User Behavior in P2P Systems. In Klaus Wehrle, Mesut Günes, and James Groß, editors, *Modeling and Tools for Network Simulation*. Springer, first edition, 2010.

[5] Konstantin Pussep, Sebastian Kaune, Jonas Flick, and Ralf Steinmetz. A Peer-to-Peer Recommender System with Privacy Constraints. In *3rd International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC)*. 2009.

[6] Konstantin Pussep, Simon Oechsner, Osama Abboud, Miroslaw Kantor, and Burkhard Stiller. Impact of Self-Organization in Peer-to-Peer Overlays on Underlay Utilization. In *4th International Conference on Internet and Web Applications and Services (ICIW)*. 2009.

[7] Konstantin Pussep, Sebastian Kaune, Christof Leng, Aleksandra Kovacevic, and Ralf Steinmetz. Impact of User Behavior Modeling on Evaluation of Peer-to-Peer Systems. Technical Report KOM-TR-2008-07, Multimedia Communications Lab, Technische Universität Darmstadt, 2008.

[8] Konstantin Pussep, Matthias Weinert, Aleksandra Kovacevic, and Ralf Steinmetz. On NAT Traversal in Peer-to-Peer Applications. In *17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. 2008.

[9] Konstantin Pussep, Predrag Knezevic, Nicolas Liebau, and Ralf Steinmetz. Improving XPath Query Execution in P2P XML Storage by Using a Decentralized Index. In *5th International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*. 2007.

[10] Konstantin Pussep, Matthias Weinert, Nicolas Liebau, and Ralf Steinmetz. Flexible Framework for NAT Traversal in Peer-to-Peer Applications. Technical Report KOM-TR-2007-06, Multimedia Communications Lab, Technische Universität Darmstadt, 2007.

B.2 OTHER PUBLICATIONS

[1] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Enabling Resilient P2P Video Streaming: Survey and Analysis. *Multimedia System Journal (MMSJ)*, 17(2):1–21, 2011.

[2] Osama Abboud, Thomas Zinner, Konstantin Pussep, and Ralf Steinmetz. On the Impact of Quality Adaptation in SVC-based P2P Video-on-Demand Systems. In *ACM Multimedia Systems Conference (MMSys)*. 2011.

[3] Osama Abboud, Konstantin Pussep, Markus Mueller, Aleksandra Kovacevic, and Ralf Steinmetz. Advanced Prefetching and Upload Strategies for P2P Video-on-Demand. In *ACM Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking*. 2010.

[4] Osama Abboud, Thomas Zinner, Eduardo Lidanski, Konstantin Pussep, and Ralf Steinmetz. StreamSocial: A P2P Streaming System with Social Incentives. In *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*. 2010.

[5] Osama Abboud, Thomas Zinner, Konstantin Pussep, Simon Oechsner, Ralf Steinmetz, and Phuoc Tran-Gia. A QoE-Aware P2P Streaming System Using Scalable Video Coding. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2010.

[6] Sebastian Kaune, Wählisch Matthias, and Konstantin Pussep. Modeling the Internet Delay Space and its Application in Large Scale P2P Simulations. In James Groß Klaus Wehrle, Mesut Günes, editor, *Modeling and Tools for Network Simulation*. Springer, first edition, 2010.

[7] Sebastian Kaune, Gareth Tyson, Konstantin Pussep, Andreas Mauthe, and Ralf Steinmetz. The Seeder Promotion Problem: Measurements, Analysis and Solution Space. In *19th International Conference on Computer Communications and Networks (ICCCN)*. 2010.

[8] Sergios Soursos, Maria Angeles Callejo Rodriguez, Konstantin Pussep, Peter Racz, Spiros Spirou, George D. Stamoulis, and Burkhard Stiller. ETMS: A System for Economic Management of Overlay Traffic. In *Towards the Future Internet - Emerging Trends from European Research*. IOS press, 2010.

[9] Osama Abboud, Aleksandra Kovacevic, Kalman Graffi, Konstantin Pussep, and Ralf Steinmetz. Underlay Awareness in P2P Systems: Techniques and Challenges. In *23th IEEE International Parallel and Distributed Processing Symposium, Workshops and Phd Forum (IPDPSW)*. 2009.

[10] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Quality Adaptive Peer-to-Peer Streaming Using Scalable Video Coding. In *12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)*. 2009.

[11] Sebastian Kaune, Konstantin Pussep, Christof Leng, Aleksandra Kovacevic, Gareth Tyson, and Ralf Steinmetz. Modelling the Internet Delay Space Based on Geographical Locations. In *17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*. 2009.

[12] Simon Oechsner, Frank Lehrieder, Tobias Hossfeld, Florian Metzger, Konstantin Pussep, and Dirk Staehle. Pushing the Performance of Biased Neighbor Selection Through Biased Unchoking. In *9th International Conference on Peer-to-Peer Computing (P2P)*. 2009.

[13] Juan Fernandez-Palacios, Maria Angeles Callejo Rodriguez, Hasan Hasan, Tobias Hossfeld, Dirk Staehle, Zoran Despotovic, Wolfgang Kellerer, Konstantin Pussep, Ionna Papafili, George D. Stamoulis, and Burkhard Stiller. A New Approach for Managing Traffic of Overlay Applications of the SmoothIT Project. In *2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS)*. 2008.

[14] Kalman Graffi, Konstantin Kaune, Sebastian Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Load Balancing for Multimedia Streaming in Heterogeneous Peer-to-Peer Systems. In *18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. 2008.

[15] Sebastian Kaune, Tobias Lauinger, Aleksandra Kovacevic, and Konstantin Pussep. Embracing the Peer Next Door: Proximity in Kademlia. In *8th IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2008.

[16] Sebastian Kaune, Konstantin Pussep, Gareth Tyson, Andreas Mauthe, and Ralf Steinmetz. Cooperation in P2P Systems through Sociological Incentive Patterns. In *3rd International Workshop on Self-Organizing Systems (IWSOS)*. 2008.

[17] Kalman Graffi, Konstantin Pussep, Sebastian Kaune, Aleksandra Kovacevic, Nicolas Liebau, and Ralf Steinmetz. Bandwidth Management: Scheduling and Active Queue Management of Overlay Flows. In *32nd IEEE Conference on Local Computer Networks (LCN)*. 2007.

[18] Kalman Graffi, Konstantin Pussep, Nicolas Liebau, and Ralf Steinmetz. Taxonomy of Active Queue Management Strategies in Context of Peer-to-Peer Scenarios. Technical Report KOM-TR-2007-01, Multimedia Communications Lab, Technische Universität Darmstadt, 2007.

[19] Nicolas Liebau, Konstantin Pussep, Kalman Graffi, Sebastian Kaune, Eric Jahn, André Beyer, and Ralf Steinmetz. The Impact Of The P2P Paradigm. In *Americas Conference on Information Systems (AMCIS)*. 2007.

C

CURRICULUM VITAE

PERSONAL

Name	Konstantin Pussep
Place of Birth	Novosibirsk, Russia

EDUCATION

Since 16/10/2006	Technische Universität Darmstadt Doctoral candidate at the Department of Electrical Engineering and Information Technology
10/2000–8/2005	Technische Universität Darmstadt Studies of Computer Science Degree: Diplom-Informatiker
6/1999	Berufliches Gymnasium Michelstadt Abitur, majoring in Mathematics and Economics

WORK EXPERIENCE

Since 16/10/2006	Technische Universität Darmstadt Research assistant at the research group *Peer-to-Peer Systems* at the Multimedia Communications Lab (KOM)
1/2006–10/2006	Fraunhofer Institute IPSI Research Assistant

TEACHING ACTIVITY

2007–2009	Lab Exercise *Communication Systems and Multimedia I/II*
2008/2009	Seminar *Advanced Topics in Future Internet Research*
2007–2010	Lecture and Exercise *Communication Networks I*
2006–2010	Tutor for various Bachelor-, Diploma, and Master theses

HONORS

12/2005	Award *Beste IPSI-Diplomarbeit 2005*

SCIENTIFIC ACTIVITIES

TPC Membership	3rd Workshop on Economic Traffic Management (ETM), 2010
Reviewer	Computer Communications, 2009 Computer Networks Journal, 2008 IEEE Wireless Communications & Networking Conference, 2009 3rd Workshop on Economic Traffic Management (ETM), 2010 ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2009 IEEE International Conference on Peer-to-Peer Computing (P2P), 2007–2010

D

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 2011

Konstantin Pussep