On Energy-Awareness for Peer-assisted Streaming with Set-Top Boxes

Konstantin Pussep, Sebastian Kaune, Osama Abboud, Christian Huff, Ralf Steinmetz
Multimedia Communications Lab, Technische Universität Darmstadt, Germany
Email: *Firstname.Secondname*@kom.tu-darmstadt.de

***Abstract*—Energy consumption is responsible for a large fraction of costs in today's content distribution networks. In upcoming decentralized architectures based on set-top boxes (STB), acting as tiny servers, idle times can dominate distribution costs, since no cooling costs occurs and the Internet access is often paid in a flat-rate manner. The often assumed always-on property of STBs provides high availability but might also waste up to 93% of the baseline energy. In this paper we consider suitable standby policies that reduce energy consumption but still allow offloading content servers significantly. We devise optimal and heuristic standby policies and evaluate them in a realistic scenario to show that a near-optimal behavior can be reached by utilizing the specific features of STBs.**

I. Introduction

Streaming of video content is a major trend on today's Internet and some forecasts predict that by the end of 2013 the equivalent of 10 billion DVDs per month will cross the Internet [1]. Due to the high costs and scalability issues of pure server-based solutions, peer-assistance is an attractive option to combine benefits of Peer-to-Peer (P2P) and server-based technologies [2]. Recent research suggests using end-users' set-top boxes (STBs), such as IP-enabled video recorders, digital receivers, or DSL modems, to build collaborative content delivery systems [3]. Two special features make them attractive: manageability of resources, such as disc space and bandwidth, and their availability, since it is often assumed that STBs are always online [4], [5], [6]. Therefore, such STB-based systems are expected to be a more suitable platform for IPTV services, including video-on-demand and live content, than traditional best-effort P2P systems.

A content provider's main goal in utilizing an STB-based delivery network is to save bandwidth and 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 [6], [7]. Users contribute their upload bandwidth and online time. As Internet access is typically paid in a flat-rate manner, the additional load might not result in additional costs for the users. On the other hand, excessive online time might result in unnecessary energy wastage, especially, if the STBs stay always online, though being idle to a large extent.

While the baseline energy consumption of STBs is assumed to be low (only 12-55 watt for typical devices) compared with desktop computers, it still results in annual rates comparable to electric ovens and refrigerators, exceeding the annual consumption of desktop and laptop computers [8]. The more powerful the STBs become, the higher energy consumption can be expected, as in the case of game consoles consuming easily 150 watt in the idle state [9].

Since typical STBs consume almost the same energy in the idle and busy mode [6], a more efficient management of online times appears promising. Assuming, that the busy time of STBs (such as DVRs) is only 100 minutes on average [6], the theoretical savings of up to $1 - \frac{100m}{24h} = 93\%$ are possible. However, naïve policies might result in high load on content servers by making the required content unavailable. Therefore, intelligent standby policies are required to provide high performance while reducing the idle times by turning STBs offline in a self-managed way.

In this paper we identify the need for intelligent standby policies for STB-based content delivery networks. Based on realistic data we assess the trade-off between the performance and energy costs of such systems. We propose an adaptive policy, which tries to save baseline energy of dispensable STBs by putting them into the sleeping mode. Trace-driven simulations are used to analyze the efficiency of the proposed policy in comparison with alternative approaches. The results show that our adaptive heuristic policy can reach performance similar to the optimal, while reducing the energy consumption significantly.

The paper is structured as follows: Section II presents the related work and our system's architecture is described in Section III. In Section IV describes the proposed standby policy and discusses alternative solutions. Evaluation methodology is covered in Section V and the results presented in Section VI. Finally, Section VII concludes the paper.

II. Related Work

Laoutaris et al. proposed to replace traditional data centers with provider-managed distributed STB networks instead of PC-based P2P systems, since the former expose higher security, Quality-of-Service (QoS), and coordination, which make them useful for live and VoD streaming [3]. Several other works deal with video streaming by utilizing set-top boxes or home gateways, focusing on the specifics of live streaming [5], [10], or video-on-demand streaming [11], [12], [4], [6]. Most works 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 [4], [5]. However, alternative deployments require energy-aware solutions, where STBs cooperate to deliver content on behalf of various content providers across domain boundaries. Here the bandwidth is not plentiful and energy costs for users are important.

A popular aspect is the pro-active 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 [12], [6]. The resulting systems "free-ride" on already dedicated baseline resources in order to reduce server load [12] or energy consumption of content servers [6]. The latter work assumes that STBs might optionally support standby mode but consider only a simple selfish policy (where users switch off the STB after finishing their downloads), instead of adaptive policies proposed in our paper. The always-on assumption might hold for home gateways (such as DSL modems with hard drives), but not for IP-enabled DVR, game consoles, and other entertainment devices.

There exist also other approaches to make the content distribution "greener". For example, Lee et al. have shown that the energy consumption of always-on STBs might be even higher than that of traditional content distribution [7]. They propose a combination of energy-proportional computing and content-centric networking with routers caching the content close to the users. To make STBs more competitive we drop the assumption that STBs must be online most of the time and show the benefits and challenges of adaptive standby policies.

III. System Architecture

Though an STB-based architecture can be used to offer various services and applications, in this work we focus on video-on-demand streaming. The considered hybrid content delivery network combines centralized and decentralized entities as depicted in Fig. 1. The architecture comprises three types of entities: indexing server, content server(s), and peers (being STBs). The system works in a cooperative manner, which means that peers are incentivized to contribute their resources, e.g. via video price discounts through the content provider.

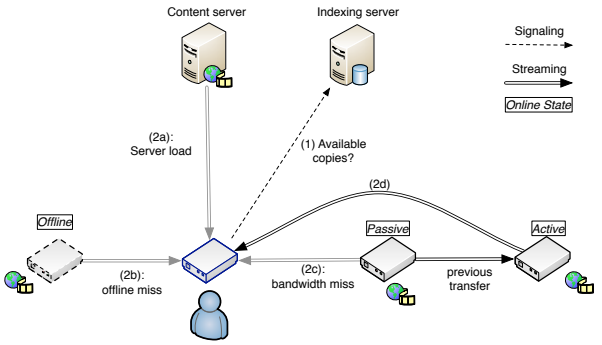


Fig. 1: System architecture. A video request results in four alternative source types, where only (a) and (d) can be successful.

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 STBs. This functionality is unavoidable to deal with unpredictable user demand but the load on these servers must be minimized to reduce hosting cost.

The *indexing server* tracks videos available in the overlay, including the information which videos has been already downloaded by peers, and provides information about available video replicas to peers requesting them. For this purpose, it collects the information about the current load, online/offline state, and cache content of single STBs. It also answers search requests with the list of peers holding video replicas. Optionally, a decentralized search mechanism (such as gossiping or DHT) 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, therefore, is not our main focus.

Finally, the *STB peers* download videos on behalf of their users' requests, store videos in (size-limited) local caches, and offer videos to other peers. As shown in Fig. 1 an STB can be in several states. It can be *active* while requesting and downloading a video from multiple sources in parallel. Hereby, it can be forwarding the same video or upload previously cached videos to other peers. In the *passive* state a peer is uploading data to other peers without downloading. It can be in the *sleeping* mode and not reply to download requests. Finally, it can be in the *idle* state while not downloading or uploading any videos but being able to serve requests from other peers. This latter state can result, for example, from the always-on behavior of STBs, and make up a major part of their online times and baseline energy consumption.

In order to download a video the requesting peer first connects to the indexing server and receives a list of peers that already downloaded this video in the past. The video can be streamed from multiple sources (peers or servers) by dividing the video into sub-streams that are combined into single stream at the receiver side. If the peers cannot satisfy the rate demand (available peers' capacity is lower than the video bitrate) the requester asks a content server to provide the missing substreams.

There are several reasons why a peer might not be able to upload previously consumed video to the requesting peer:

1) The peer might have already removed the video from its local cache (cache miss).
2) The peer might be offline (offline miss).
3) The peer might not have free upload capacity because it is already uploading to other peers (bandwidth miss).

Since the main goal of the system is to offload the content servers as much as possible, different mechanisms are applied 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. For example, a cache replacement policy is used to reduce caches misses while a proper peer selection mechanism is applied to avoid bandwidth misses and achieve fair load distribution.

We further aim to reduce the energy consumption of STBs in the system by applying a *standby policy* that governs when idle peers can switch into the sleeping mode and back. This policy is used to 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 load. We discuss possible policies in the next section.

IV. Adaptive Standby Policies

As explained above, standby policies govern the switching between the idle and sleeping modes of an STB. A properly designed policy should fulfill the following requirements:

1) Assure that the content available in the overlay can be served from peers (avoid offline misses).
2) Minimize the idle time of peers by sending them into the sleeping mode whenever their services are not required.

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 of the system.

A. Optimal Policy

This policy switches idle STBs into the sleeping mode immediately after finishing downloads (and currently running uploads). If another peer tries to download a video and the online STBs cannot provide the required video bitrate, the indexing server computes the missing bandwidth and selects one or several sleeping peers owning a replica of the desired video. The policy *wakes up* those peers, which then provide the desired bandwidth and switch back into the sleeping mode after finishing the upload. We expect this policy to exhibit the optimal performance since it assures that *all required* peers are available when they are needed while eliminating idle times altogether.

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[1].

Several issues arise with this policy, besides the required support of the Wake-on-LAN feature:

- Firewalls and routers might drop the wake-up packets as a potential security threat.
- STB's wake-up time might delay the video startup for the requester by tens of seconds.
- The policy cannot be applied if the STB itself, such as DSL modems, must be running for Internet connectivity since the Wake-on-LAN requires the STB to be reachable over the Internet.

Therefore, we need an alternative policy that is able to approximate this behavior without relying on the Wake-on-LAN feature.

B. Popularity- and Supply-aware Policy (PSP)

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 mode for the pre-defined amount of time. Such timers are common for digital video recorders, digital receivers etc., and can be incorporated into other kinds of STBs without the security limitations and wake-up delays of the wake-up policy.

The idea behind this policy 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. To do so an STB analyzes the global availability

[1] http://www.wakeonlan.me/

of its cached content in regular intervals, both in the idle and in the sleeping state. In the latter case the STB is switched on by the timer 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.

In the following we present the decision metric: Once a peer p has to make a decision whether to switch (or stay) into the *idle* state or not, it calculates *desirability* of each cached file f as follows:

$$D(p,f) = max\Big(s \cdot R(f) - A(f,p), 0\Big) \cdot H(f,T)$$

Here $R(f)$ is the bitrate of the video, s is the required *supply factor*, and $A(f,p)$ is the total upload rate available at all peers holding a replica of f, except p itself. The latter upload availability includes only unused upload bandwidth of online peers. $H(f,T) \rightarrow \{0,1\}$ specifies whether video f was requested by any other peer within the last T hours or not.

With $F(p)$ being all videos cached at peer p the peer's desirability is then:

$$D(p) = \sum_{f \in F(p)} D(f,p)$$

Intuitively, this metric reflects whether the locally cached files are sufficiently replicated in the overlay. The limited history considered by the metric allows avoiding STBs staying online to offer old and unpopular videos.

Only if the computed desirability is equal to zero the STB goes into the sleeping mode for the next hour. Otherwise it stays online for at least one hour. After this interval the STB repeats the computation. One of the benefits compared to the wake-up policy is that this policy can avoid the startup delay if a peer must be awaken, which is crucial for delay-sensitive streaming applications.

A proper configuration of the policy parameters T and s should allow *PSP* to achieve performance close to the wake-up policy and outperform alternative policies. The information about available video replicas and recent requests can be either managed by the indexing server, or distributed in the overlay via gossiping. Note that too low supply factor s might result in high load at content servers, while higher values of s will increase the online time of STBs. On the other hand, the history length T is important to react on content popularity changes.

C. Alternative Policies

For the purpose of completeness, we also consider alternative policies: We consider the *always-on* behavior as one extreme policy, where an STB stays online once it joins the system. Due to the permanent availability of all peers, such a policy allows for the highest possible peer contribution, and, in consequence, highest load reduction at content servers. On the other hand, 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 content provider to the end user.

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, i.e, watch a video. Once a user finishes its downloads it switches off the STB and its resources become unavailable to the system. Though the peers can contribute their upload resources while they are online, we expect that such a policy results in much higher server load than the always-on policy. On the other hand, this policy results in a minimal idle time, in fact being zero.

The *overtime* policy incites or forces users to keep their STBs online even if not using them actively. Here STBs stay online for a fixed *seeding time* (e.g., one hour) after finishing their downloads. The performance of this policy strongly depends on the concurrency of user requests: if most users perform requests almost in parallel, moderate seeding might suffice to achieve high server load reduction. However, the non-adaptive nature of this policy might also result in unnecessary idle times or high server load.

Table I shows an overview of considered policies, together with their ability to fulfill the presented requirements. After discussing each policy (including its benefits and drawbacks) in detail we will evaluate them in Section VI to verify our expectations.

TABLE I: Expected performance of considered policies.

Policy	Server load	Idle times	Remarks
Always-on	minimal	high	
Selfish	high	minimal	
Overtime	Depends on seeding time and request pattern		
Wake-up	minimal	optimal	Requires Wake-on-LAN
PSP	moderate	near-optimal	Requires timer

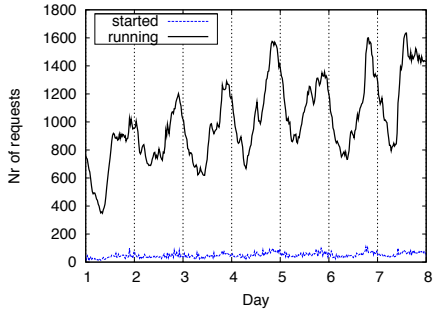
V. Evaluation Methodology

This section presents evaluation methodology, including the workload, simulation environment, and metrics.

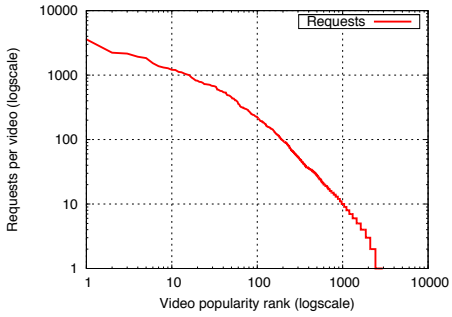
A. Workload

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

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 [13]. 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.



(a) User request pattern



(b) Movie popularity

Fig. 2: Properties of trace content.

Fig. 2 shows an excerpt of the traces, obtained by taking a random sample of peers that downloaded content from the category *Movies* during the first week of measurements (Dec. 13-21, 2008). It can be seen that the trace shows a strong similarity with typical VoD workloads: the requests over time follow a clear diurnal patter (similar to e.g. [2, Figure 7]) and the video popularity follows the stretched exponential distribution of typical multimedia workloads [14].

The basic scenario used for the performance evaluation is shown in Table II. We use 8 continuous days out of our traces, while the first day is used to warm-up the system and does not contribute to the metric computation. The minimal size of considered videos is

TABLE II: Basic setup.

Parameter	Default	Variation
Simulation duration	8 days	
Video bitrate	1, 2, 4 Mbps	
Peer cache size	2 GB	
Peer upload bandwidth	1 Mbps	0.256 – 4 Mbps
Number of peers	60,000	10,000 – 60,000
Number of videos	2,620	1,260 – 2,620
Standby policy	always-on	selfish, overtime, wake-up, PSP

200 MB and we derive the bitrate of the video depending on the video size S according to the following rule: $S \leq 1GB \rightarrow 1Mbps$, $1GB < S \leq 2GB \rightarrow 2Mbps$, and, finally, $S \geq 2GB \rightarrow 4Mbps$ bitrate. The default peer upload bandwidth is set to 1 Mbps. 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** Mio. peers for the selected content category *Movies* for the considered week).

B. Simulation Environment

The traces are fed into a custom discrete event-driven simulator modeling a peer-assisted streaming system based on set-top boxes. We implemented the delivery network presented in Section III with the users consuming videos and overlay provider's servers offering the initial content. 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 IV.

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.

C. Metrics

We use the following metrics to evaluate the performance of the policies:

- *Overlay hit rate* HR is the data volume served by peers relative to the totally served data. This metric describes the *performance* of the system, since the higher hit rate at STBs means higher benefit for the content provider.
- *Online time* OT is the accumulated online time of all STBs relative to the always-on case. 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 energy consumption dominates the energy costs of STBs [6]).

- *Energy savings* S express the relative reduction in STBs online time achieved by policy P compared to the always-on case. We further normalize this value by the hit ratio of P compared to the always-on policy A for a fair comparison, i.e., $ES(P) = \frac{OT(A)-OT(P)}{OT(A)} \cdot \frac{HR(P)}{HR(A)}$

VI. 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 goal of our experiments is two-fold: to compare the performance and costs of standby policies and to study the impact of policy-internal parameters, such as supply factor and history length for PSP. Hereby, we want to verify our policy analysis summarized in Table I.

A. Comparison of Standby Policies

In the first experiment, we compare the performance of PSP with that of other standby policies: always-on, wake-up, overtime (seeding time = 1 hour) and selfish. Figure 3 shows the overlay hit ratio, the relative online time, and the energy savings (except for the always-on policy).

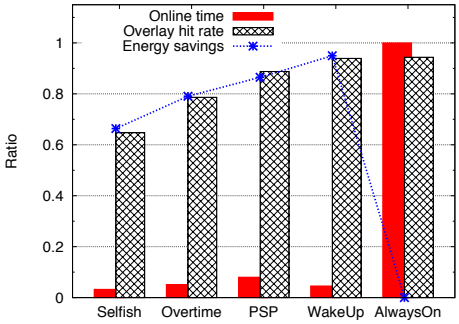


Fig. 3: Overlay hit rate, online time, and energy savings of various policies.

We observe that our heuristic policy, PSP, results in an overlay hit rate close to the optimal wake-up policy (89 vs. 94%) while slightly increasing the online time (8 vs. 4.5%). Consistently the energy savings are similar (86 vs. 92%) as well. On the other hand, *PSP clearly outperforms the selfish and overtime policies regarding the overlay hit rate*, and always-on policy regarding the online time. Furthermore, the always-on and wake-up policies offer the highest overlay hit rates of 94% with the remaining 6% accounting for the initial content injection from the content servers. Selfish and always-on policies result in the lowest and highest online times, respectively.

In the next experiments we consider the impact of various system and policy parameters on the performance of standby policies. Since the normalized energy savings show the trade-off of performance and costs we will focus on this metric for the workload parameter study.

1) Impact of Upload Bandwidth: We evaluate the impact of the available peer upload bandwidth and compare the energy savings for each policy relative to the always-on behavior (see Fig. 4). 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 3 Mbps, since here each peer can upload 1 or 2 complete streams in parallel (average video bitrate is below 2 Mbps). But still the PSP is able to save almost 10% more energy than the non-adaptive selfish and overtime policies, coming quite close to the optimal wake-up policy.

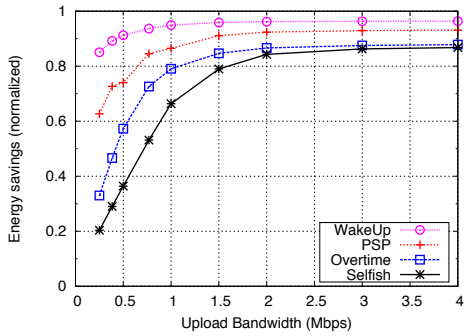


Fig. 4: Energy savings compared to the always-on behavior (normalized by the overlay hit rate) with varying upload bandwidth per peer.

For lower upload capacities the difference between the policies is much higher, e.g, with 256 kbps per peer resulting in more than 60% savings with the PSP but only 20% with the selfish policy. In such a low-dimensioned scenario, streaming of a single video requires 4-12 online STBs to cover the demand.

2) Impact of Population Size: In our next experiment we analyze the impact of the population size. For this purpose we generate random subsets with the different number of peers (10,000 to 60,000) and investigate the energy savings (see Fig. 5). We observe that for most policies *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 (22.9 vs. 7.9 requests for 60,000 and 10,000 peers). This yields a higher replication degree,

which allows more peers to go offline without hurting the overlay performance. While other policies are able to improve their performance, *the wake-up policy is again the best one and PSP offers a good trade-off outperforming the overtime and selfish policies*.

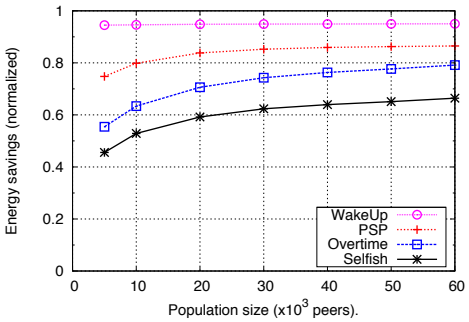


Fig. 5: Energy savings compared to the always-on behavior (normalized by the overlay hit rate) as a function of population size.

B. Analysis of Selected Policies

The previous experiments compare the performance of best (empirical) parametrization of standby policies and their scalability regarding the system parameters. As next we perform a parameter study of the overtime policy and heuristic PSP. The goal is two-fold: to understand the impact of intrinsic parameters and to verify whether the adaptive PSP can outperform the non-adaptive overtime with the best parametrization (i.e., optimal seeding time). For this purpose we vary the relevant parameters of each policy and measure the impact.

1) Overtime Policy: In this experiment we evaluate the potential of increased seeding time for the overtime policy. This should allow us to estimate the potential of this *non-adaptive* approach and to compare it with the PSP. Figure 6 shows the impact of the increased seeding times per peer on the overlay hit rate, relative online times, and normalized energy savings. We observe that *higher seeding time up to the value of 3 hours increase normalized energy savings*. Above that level of three hours the normalized savings are continuously decreasing because the required online time grows faster than the overlay hit rate.

This effect can be explained by the common flash-crowd effect often observed in content distribution networks [14]. Once a new video becomes available, a lot of concurrent requests are started. If the peers, which download the video first, stay in the network for the duration of the flash-crowd, they can serve the newcomers. After these demand peaks the longer seeding becomes unreasonable. This can be aggravated by the diurnal traffic pattern, which typically has its load peaks in the few evening hours (cf. Fig. 2a).

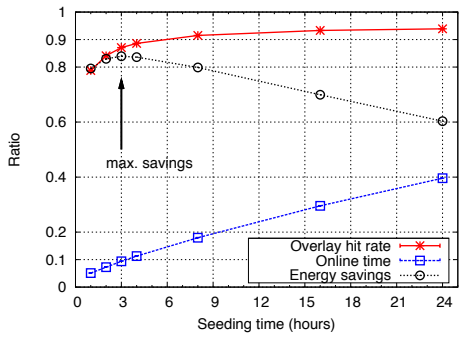


Fig. 6: Impact of seeding times on the *overtime* policy performance.

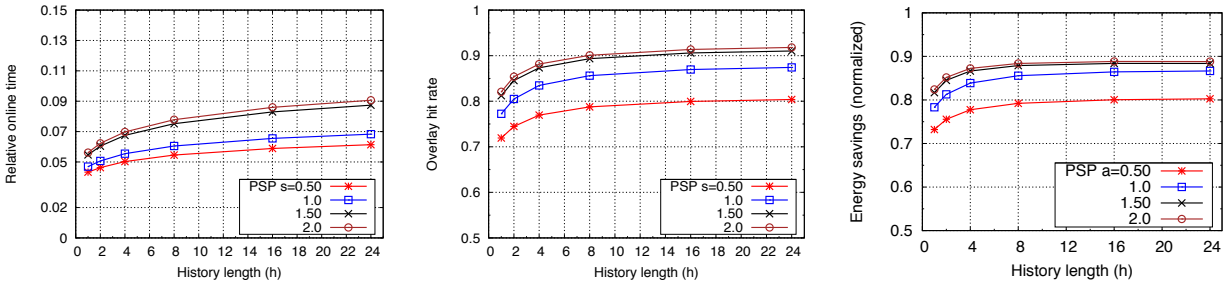
2) Popularity- and Supply-aware Policy: In this experiment, we consider our PSP policy and vary its main parameters: *history length* from 0 to 24 hours and *supply factor* between 0 and 2 (see Fig. 7). We observe that:

- Online time increases linearly with the history length while even with increased supply factor the total fraction is below 10% (see Fig. 7a).
- The overlay hit rate exceeds the 90% level with the supply factor of 2.0 and history length ≥ 8 hours (see Fig. 7b).

The reason is that taking into account longer history makes peers stay online to offer their content longer, while the supply factor ≥ 1.0 allows serving at least one new request by the peer kept online.

Furthermore, the relative energy savings (shown in Fig. 7c) increase with the history length and higher supply factor but stabilize at the supply factor of 2.0 and history length of 16 hours. Here the increase of the supply factor from 1.5 to 2 results only in marginal savings increase. These observations suggest that further increase in supply factor and history length is not beneficial.

3) Overtime vs. PSP: In order to compare the performance of the overtime policy and PSP, in Fig. 8 we plot the overlay hit rate vs. online time for both policies with varying policy-internal parameters (the impact of varied parameters in indicated by the arrows). The figure visualizes that *PSP with the supply factor* ≥ 1.0 *outperforms the overtime policy regarding both the system performance (overlay hit rate) and costs (relative online time).* The reason is the adaptive nature of the PSP, which adjusts the online sessions to the demand and supply, while overtime policy statically tries to find the "one size fits all" seeding time.



(a) Relative online time (b) Overlay hit rate (c) Energy savings (normalized)

Fig. 7: Impact of history length h and supply factor s on the PSP's performance.

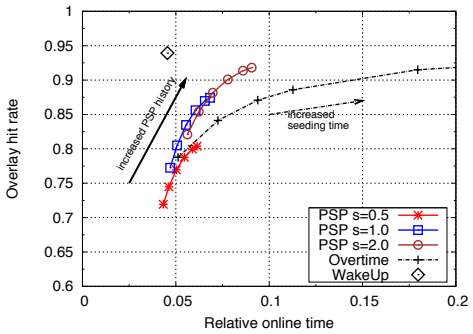


Fig. 8: Comparison of *performance* and *costs* with varying parameters: for PSP the supply factor is fixed, but the history length variable (1-24h), while for the overtime policy the seeding time is varied (1-8h).

VII. Conclusion

This paper presents an adaptive standby policy to reduce energy consumption of a content delivery network based on set-top boxes. This is done by reducing *idle* times. The set-top boxes can switch into the sleeping mode and back in a self-managed way to reduce the baseline energy consumption. To achieve this, we devise optimal and heuristic policies and evaluate their performance. Our results show that adaptive standby policies can tackle the trade-off between the extensive online time and high overlay hit rate, resulting in similar performance as the always-on behavior at the slightly higher costs than the selfish policy. The proposed policies also outperform non-adaptive overtime behavior.

Acknowledgements

This work has been partially supported by the EU ICT Project SmoothIT (FP7-2007-ICT-216259) and by the German Federal Ministry of Education and Research (BMBF) in the project Premium Services (01IA08003A).

The authors would like to thank all project partners for useful discussions on the subject of the paper. We also thank Thorsten Strufe for his valuable feedback.

References

[1] "Cisco Visual Networking Index: Forecast and Methodology, 2008-2013, White Paper," 2009.

[2] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *SIGCOMM '08*, vol. 38, no. 4, 2008, p. 375.

[3] N. Laoutaris, P. Rodriguez, and L. Massoulie, "ECHOS: edge capacity hosting overlays of nano data centers," *SIGCOMM Computer Communication Review*, vol. 38, pp. 51–54, 2008.

[4] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-Peer Video-on-Demand system: design and evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, p. 1706, 2007.

[5] J. He, A. Chaintreau, and C. Diot, "A performance evaluation of scalable live video streaming with nano data centers," *Computer Networks*, vol. 53, pp. 153–167, 2009.

[6] V. Valancius, N. Laoutaris, L. Massoulie, C. Diot, and P. Rodriguez, "Greening the Internet with Nano Data Centers," in *CONEXT*, 2009.

[7] U. Lee, I. Rimac, and V. Hilt, "Greening the Internet with Content-Centric Networking," in *e-Energy*, 2010.

[8] N. Horowitz, "NRDC Study of Set Top Box and Game Console Power Use," 2007. [Online]. Available: http://www.energystar.gov/ia/partners/prod_development/revisions/downloads/settop_boxes/NRDC_SetTopBox_Data_IEA.pdf

[9] E. Bush, M. Schalcher, P. Kühne, S. Kammermann, S. Gasser, and J. Nipkow, "Measurement of the power consumption of set-top boxes," 2007. [Online]. Available: http//www.topten.ch/uploads/images/download-files/Schlussbericht-Settop-Boxen-V14_EN2-total.pdf

[10] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft, "On next-generation telco-managed P2P TV architectures," in *International workshop on Peer-To-Peer Systems (IPTPS)*, 2008.

[11] V. Janardhan and H. Schulzrinne, "Peer assisted VoD for set-top box based IP network," in *P2P-TV '07*, 2007, pp. 335–339.

[12] Y. Chen, R. Jana, D. Stern, B. Wei, M. Yang, and H. Sun, "Zebroid: using IPTV data to support peer-assisted VoD content delivery," in *NOSSDAV*, 2009.

[13] S. Kaune, G. Tyson, K. Pussep, A. Mauthe, and R. Steinmetz, "The Seeder Promotion Problem: Measurements, Analysis and Solution Space," in *ICCCN*, 2010.

[14] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang, "The stretched exponential distribution of internet media access patterns," *ACM PODC '08*, p. 283, 2008.