Peer-to-Peer Netw. Appl. (2010) 3:351–362
DOI 10.1007/s12083-009-0063-5

An experimental analysis of Joost peer-to-peer VoD service

Jun Lei · Lei Shi · Xiaoming Fu

Received: 17 March 2009 / Accepted: 22 September 2009 / Published online: 6 October 2009

Abstract Despite strong interest in peer-to-peer (P2P) Video-on-Demand (VoD) services, existing studies mostly focus on peer-to-peer or overlay protocol design based on simulations under various topological constraints. We believe experimental studies on a real-life P2P VoD system will provide valuable information to ISPs, network administrators, and content owners. In this paper we present a comprehensive analytical and experimental study on Joost, one of the first commercial P2P VoD systems used for distributing various forms of video over the Internet. Our extensive experiments prove that Joost is a server-assisted peer-to-peer VoD system. With several envisioned typical scenarios we have further investigated the peer management in terms of time pattern, bandwidth consumption and locality considerations. Our major findings include: (1) the current Joost system is capable of providing high-quality VoD service through the use of an overlay network deployed with a set of centralized content servers; (2) inter-continental links are often used regardless of the number of local users, which may pose a high burden on the network providers; (3) easily reachable, high-capacity nodes are selected as main relaying nodes, similar to super nodes in Skype, to facilitate the traversal of symmetric NATs and firewalls. We also provide insights on the potential ways to construct more efficient P2P VoD systems (e.g. considering topological locality-awareness, using adaptive/layered video).

Keywords Peer-to-peer (p2p) · Video-on-demand (VoD) · Measurement

J. Lei (✉) · L. Shi · X. Fu
Institute of Computer Science, University of Göttingen, Göttingen, Germany
e-mail: lei@cs.uni-goettingen.de, lei@informatik.uni-goettingen.de

L. Shi
e-mail: shi@cs.uni-goettingen.de

X. Fu
e-mail: fu@cs.uni-goettingen.de

1 Introduction

In the recent few years, IPTV has gained a tremendous popularity in operators and users as well as a lot of attention from the research community [9, 11, 12]. For residential users, such service is often provided in conjunction with VoD and may be bundled with other Internet services such as Voice over IP (VoIP). Traditionally, when a user selects a program, a point-to-point unicast connection is established between a decoder (aka *set top box*) and media server, which lacks efficiency and scalability. Most existing VoD services mainly rely on content distribution networks (CDNs) [16] or local streaming proxies to increase system scalability as well as to alleviate the delay experienced by end users. However, their system performance and deployment still faces a key challenge as the number of users increases. Especially, if a flash crowd [14] occurs, servers can be easily overloaded. The similar phenomenon occurs when a web site catches the attention of a large number of people, and gets an unexpected volume and possibly overloading surge of traffic.

To address above issues, peer-to-peer technologies (e.g. swarming [7]) have been recently employed to support VoD services. However, it is more challenging to design a P2P VoD system than any other P2P media

streaming systems because, in addition to providing low playback delays, the system allows users arriving at arbitrary time to watch videos. The heterogeneous arrivals reduce sharing opportunities and increase the complexity of video distribution mechanisms. Besides, the system requires a certain local space to store the downloaded video. Thus, another issue is how to allocate and use such storage in an efficient way to support VoD functionalities. As current P2P VoD systems have not been widely deployed, it is essential to understand how to design a VoD architecture that scales smoothly to support a large number of users, while maintaining high video quality and reasonable operational costs. It is also critical for ISPs, network administrators, and content owners to consider the network and operational requirements for supporting P2P VoD systems [22].

Most P2P VoD studies use simulations under various common assumptions to evaluate their designs. However, due to a lack of large-scale deployed VoD systems, few of these assumptions could be validated through real measurement data. Our analysis on Joost seeks to validate and adapt existing assumptions about P2P VoD data. We choose Joost as the target for the study of peer-to-peer VoD services due to the following reasons. As one of the earliest and large-scale commercial P2P VoD products, Joost has the potential to become popular following a successful story of Skype. The Joost architecture and many technologies it uses are proprietary although it is known to be built on top of several open software such as Mozilla/xulrunner [1]. Except for the limited knowledge of the used open software, the underlying P2P architecture and detailed mechanisms/techniques used in Joost, like when Skype was new, are still unrevealed. While getting deep insights into various aspects of Joost has been challenging, our results may provide valuable information to ISPs, network administrators and content owners for a better understanding of service requirements for building and managing a scalable and efficient P2P VoD system.

The rest of the paper is organized as follows. Section 2 briefly introduces the overall architecture and identifies the functionalities performed by the components of the Joost system. Section 3 describes our experimental setup. As the performance aspect plays an important role in P2P VoD acceptance, in Section 4 we envision three typical usage scenarios and use them to study more closely on the peer behavior and service performance in terms of time pattern, bandwidth consumption and locality considerations. Section 5 presents related works. In Section 6 we conclude this paper and plan future work.

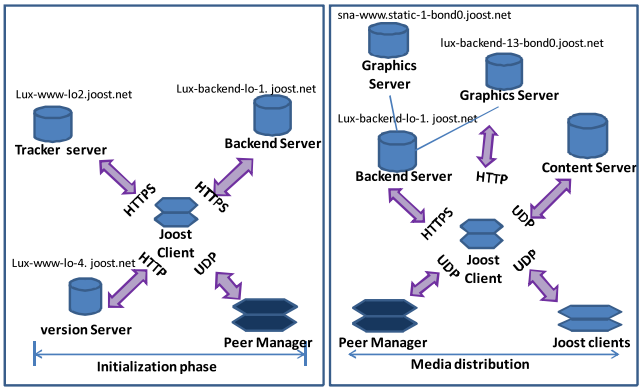

Fig. 1 Joost architecture

2 Joost overview

Joost [1], created by N. Zennström and J. Friis, co-founders of Skype [21] and Kazaa [17], is one of first P2P VoD systems for providing high-quality and comprehensive VoD services using P2P TV technologies. Based on our preliminary experiments [18] using a testbed depicted in Section 3 and information in [2], we deduced the Joost system architecture using a top-down approach: we firstly abstracted a high-level hierarchy from the overall architecture and then investigated the functionalities performed by each component of the P2P hierarchy.

Figure 1 shows five types of servers (cf. Section 2.1), one Joost peer manager,[1] and various Joost clients. There are other servers taking charge of value-added services, for example, instant chat service (scd.joost.com) and advertisement server (lux-cdn-lo-4.joost.net). Because the fundamental functions are our focus, these additional servers have been omitted from the subsequent discussion.

2.1 Joost servers

Joost is rather a complicated system, and identifying the Joost servers facilitates our understanding of the peer management mechanisms because their behaviors can be differentiated from that of arbitrary peers. In this paper, we use the term of peer and client interchangeably.

As shown in Fig. 1, lux-www-lo4.joost.net is **version server** that is responsible for checking the current version of the software during login. For instance, Joost

[1] Although peer manager is named as super node in [2], in our experiments we identify that its functionality is peer management. It is not responsible for relaying/forwarding media data to other peers. Therefore, we call it peer manager in this paper.

Clients (JCs) sent HTTP 1.1 GET requests for getting the latest software version. The second type of server is called **tracker server** (i.e. lux-www-lo2.joost.net) whose sole responsibility is to keep track of its group members and helps bootstrapping new peers.

Channel management in Joost differs largely from any other P2P VoD system as it builds an API for the channel list using XULRunner that provides Joost clients more interactive experience but makes the system more complex. Due to the complexity, the channel management is not performed by a single server in Joost, but a server cluster. That is, the **backend server**, lux-backend-lo-1.joost.net, answers for controlling channel list requests and keeping load balance among cluster servers, whereas other cluster servers perform particular tasks (e.g. channel graphs downloading). Some of them can be named channel **graphics servers**. For instance, there were two servers, lux-backend-13-bond0.joost.net and or sna-www.static-1-bond0.joost.net from which JC downloaded the channel graphics instead of directly from the backend server. Nevertheless, the scalability might be a major concern for the future development if the number of users dramatically increases in a short period.

The last type of Joost server is **content server**. During our experiments, we observed the following server sites: (1) 4.71.105.0/24 (sna-Itsnode-*x*-bondx-*x*.joost.net); (2) 4.71.174.0/24 (IPsoft); (3) 212.187.185.0/24 (Icy-Itsnode-*x*-bondx-*x*.joost.net). Here, x varies from 0 to 10. The first and third IP address site is owned by Level 3 Communication INC [19] which has been selected by Joost to support on demand Internet TV. The second IP space group belongs to IPsoft service provider.

2.2 Peer manager

Another major distinguished design from other P2P networks (e.g. Skype) or overlay multicast solutions [6] is that all peer managers in Joost are only used for controlling and helping new peers find available contributing peers. Based on above understanding, the control traffic from peer managers can be easily differentiated from other media data traffic in Section 4.3.

In fact, it is quite efficient and reasonable that the peer management is isolated from the media distribution. According to [20], some universities have already banned Skype from their campuses, while some other universities and government agencies require that their users disable supernode functionality to avoid relaying traffic outside the stub Autonomous Systems (ASs). Apparently, Joost designers have taken the issue into consideration. Moreover, strategically deployed peer managers not only ease the membership management but also improve the reliability of transmission. For example, if a super node in Skype leaves ungracefully, all the other peers relying on it will be unavoidably affected.

2.3 Joost client

While active a Joost Client (JC) performs one of the following actions: listen on particular ports for incoming traffic; store media data into its local cache; maintain a table of other peers called a host cache, uses Advanced Video Codec (AVC); determine if it is behind a NAT or firewall; and functions required by additional features, such as instant messaging. Without a surprise, Joost and Skype have some p2p mechanisms and techniques in common. Detailed information can be found in [18].

2.4 Protocols

In order to evaluate the efficiency of video transmission it is necessary to identify the protocols used for control traffic and media data traffic.

Table 1 depicts the main protocols used in the Joost. All video packets are encoded in UDP and the size is exactly 1104 bytes. Using UDP for video transmission is not reliable, however, it's quite cost-effective and time-efficient especially for large-scale peer-to-peer networks. Besides the media transmission, peers frequently communicate with each other by sending UDP probes (64 bytes). Since April 2007, when port number 4166 was assigned by IANA as the official UDP port used for Joost, all media data and some control messages (e.g. peer management) are sent from Joost servers through 4166. Tracing these specific port numbers facilitates our following performance experiments.

Table 1 Main protocols in the Joost system

Protocol	Functionality	Packet size
UDP	Video distribution	1104 bytes
	Content probe (peer to peer)	~ 64 bytes
	Channel switching	< 1000 bytes
	VoD interactions	< 150 bytes
HTTP	Software version	
	Client → server	~ 64 bytes
	Server → client	< 500 bytes
	Channel management	
	Client → server	~ 64 bytes
	Server → client	<= 1518 bytes
HTTPS	Administrative management	
	Client → server	64 bytes
	Server → client	< 500 bytes

HTTP is used for checking software version and updating the channel list during bootstrapping and initialization phases. When the JC browses the channel category, the channel graphs are downloaded in real time through HTTP from the graphics servers.

When the JC re-connects to the Joost system, HTTPS is used for the administrative management duties which include checking software version, channel list updating, obtaining trackers.

2.5 Local video cache

So far, we identified the Joost architecture and associated key components, which are most relevant to our analysis in Section 4. As supporting VoD functionalities requires a local storage, different cache management strategy has a great impact on peer management. We did some experiments to identify its impacts.

It has been observed that during the first three months of our experiments, one peer's the cache size grew up to 3.52GB (FAT 32) and has not been dwindled, however, in January 2008 Joost changed the design to clean up the local cache after rebooting the client. On the one side, such a change is rational, otherwise, the user's resources will be significantly occupied if the JC continuously switches to different channels. On the other side, it largely reduces the resource sharing opportunities among peers since newcomers cannot get video data from those who have downloaded video in their local cache during the past logins.

JCs store the media data in their local caches as "anthill_cache" [18]. Our initial assumption was that the local cache did completely store the played video and thus the JC should watch the old program directly from the local cache. However, once we disabled the Internet connection, the program surprisingly stopped, even if the same program has just been watched. Therefore, although media data has been stored locally, playback still requires a kind of codec from the remote server or an encryption key (e.g. AES key) authorized by the Joost server to access the video file. To prove the conjecture, we performed following experiments.

We launched a new channel and at that moment, the local cache was empty. After the whole channel was watched, the size of cache file grew up to 1.7 GB and the average download speed was 518 kbps. When we switched channel and then watched the same channel, the download speed dramatically dropped down to 11 kbps. Moreover, the size of local cache increased only 0.1 GB (< 6.0%) during the second watching period. It is very likely that the slightly increased cache now contains either a type of codec or an encryption key to allow the client to access the cached video file.

3 Experiment setup

There are three main Joost server sites: the USA, Europe and Asia [2]. Since all mechanisms and technologies are assumed to be used in the same way, our experiments explore European site also due to the authors' location in Germany (GMT+01:00).

All experiments were performed between September 2007 and February 2008, in which we used Windows XP SP2 machines at various geographical locations. Note that we also used Windows Vista and MAC OS X machines for the testing, however, their results haven't shown significant difference from the following results. To illustrate our major findings, two University nodes configured with public IP addresses, another two University nodes located behind a NAT/Firewall (N/F-behind nodes); and two ADSL users were selected to set up an experimental testbed. As depicted in Fig. 2, university nodes were equipped with the same processing power and connected to 100 Mbps full-duplex university LAN. ADSL users were supported with 2 Mbps download and 1 Mbps upload bandwidth. Since December 2007, Joost has been open to public although it was still called Beta version (Beta v1.0) during the time of our experiments.

For data collection, we used Wireshark [5] and Omnipeek [4]. Tools like WhereIsIP [23] were used to perform reverse country, city and ISP lookups for an IP address when Omnipeek failed to return a DNS PTR record.

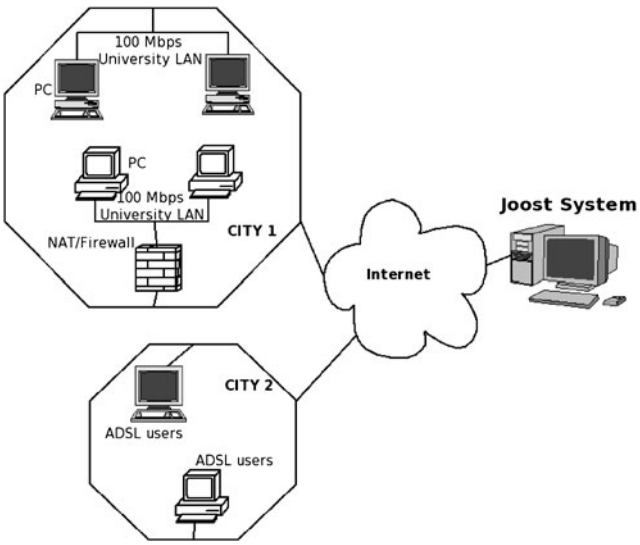

Fig. 2 Experimental testbed design

4 Inferring Joost peer management

Through three envisioned scenarios, we investigated the P2P management mechanisms which are the most important and complicated aspect in the Joost system. We resort to passive measurements and data-driven analysis in order to reveal the aspect that a purely active methodology alone cannot capture.

4.1 Experimental methodology

To facilitate our measurements, we analyze the major factors in which the peer management may involve.

- *Time pattern:* The different user distribution during a day or a week may have great impacts on the performance (e.g. contributions from peers highly depend on the number of peers).
- *Upload and download capacity:* The peer management can be benefit from the efficient bandwidth usage if peer is given some incentives to contribute more to the network.
- *Popularity impacts:* The number of users may be largely determined by the popularity of the on-demanded programs.
- *Locality considerations:* One of the main challenges in P2P VoD system is the efficient allocation of the available resources. Thus, it is generally desirable that data exchange be made preferably between nodes that are placed "close by" in the underlying network to reduce the redundant usage of long-haul network links and to save local resources for network providers.

4.2 Designed scenarios

- *Scenario 1:* To get a broader view of the time pattern, we monitored public nodes and NAT nodes over a period of three weeks (7–28 January, 2008) and captured over 78 GB data. Those test nodes were equipped with the same processing power and bandwidth support as described in Section 3. To alleviate the popularity impacts, we repeatedly ran a 1063-min channel created by our own at both nodes, including popular and unpopular programs selected from existing program list. Note how to create a new channel is out of scope of our paper, but can be found in [1]. Once the channel was finished playing, we dumped their local cache (c.f. Section 2.5) and re-started playing.
- *Scenario 2:* We randomly chose programs ranking in the "most popular programs" list and unpopular programs with the same length. In this experimental scenario, a public node and a N/F-behind node continuously ran popular and unpopular programs over two-week period (14–28 January, 2008). We captured 24 GB data.
- *Scenario 3:* Two test nodes were located behind a Network Address Translator (NAT) and configured with non-routable, private IP addresses. One of them started to randomly choose one channel. After a short period (e.g. 5 minutes), the other node selected the same channel.

Scenario 1 is mainly designed to explore above two factors, namely, *time pattern* and *upload and download capacity* in Section 4.1. The second scenario is used to investigate whether *popularity impacts* have been considered in Joost's P2P management mechanisms. In the last scenario, it would be interesting to see if the *locality-awareness* has been carefully considered in Joost system. As two NAT nodes were physically located next to each other, it would be possible that the second test node receive a large portion of data from the first node.

As media data is encapsulated in UDP (cf. Section 2.4), we only captured UDP packets and isolated media data from other control traffic. Nevertheless, Joost encrypts all UDP payloads and therefore, our analysis on the collected data is restricted to IP and UDP header fields including the source and destination IP address and port, and packet lengths.

4.3 Experimental results

Our initial experiments [18] indicated that Joost relies on a plenty of dedicated infrastructure nodes (e.g. content servers) to distribute video. However, since December 2007 during our experiments, the contributions of peers largely increased and varied according to the time/life pattern. The following experiments illustrate our recent findings.

4.3.1 Time pattern—NAT/FW-behind node

Figures 3 and 4 illustrates typical segments of first scenario results of N/F-behind node. They plot the average percentage of media contributions from Joost servers (cf. Section 2.1) and peers. We separate weekday trace from weekends trace since they have different distributions.

The first main observation from above two figures is that Joost servers delivered a majority portion of media data to Joost clients during the entire week (85% in average for weekday trace, 91% in average for weekends). The deviations identify that both traces

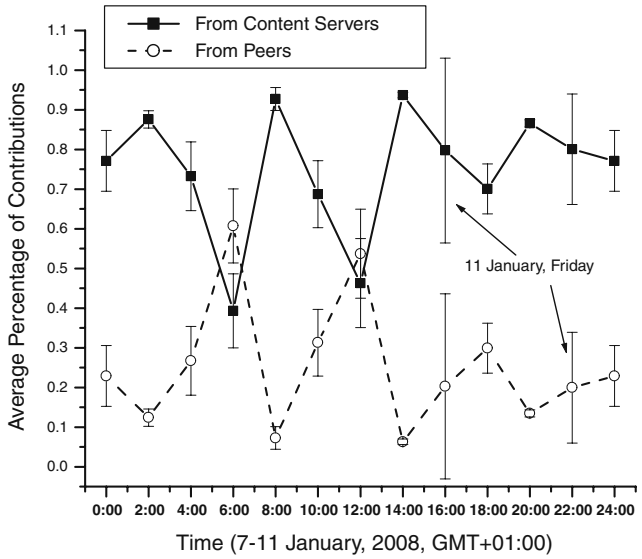

Fig. 3 Weekday trace of NAT/FW-behind node

follow the similar pattern except for two time slots in the weekdays (16:00 and 22:00, 11 January, Friday) when there were a great number of peers contributed to the test nodes. In fact, it is understandable since by that time the weekends had already started in some European countries. Note that our experimental location was in Germany. To verify our conjecture, we further traced these contributed peers at these two particular time slots. Among the total peer contributions (63.3% in average), 38% of the media data was contributed by European peers and 25.3% was transmitted from the US.

The second interesting observation is that there were two user peeks in weekday traces, 6:00 and 12:00, when a lot of contributions (50–60%) were from other peers instead of content servers. According to what has been observed in [10] that the number of users drops gradually during the early morning and climbs up to a peak when users are in noon break, the number of European users is expected to be the least in both time slots, namely, 4:00–6:00 and 10:00–12:00 in our experiments. In other words, if the contributors were located in Europe, it would be against the daily life pattern since the first time slot (4:00–6:00) will be sleeping time and 10:00–12:00 is working time. In contrast to the weekday trace, during 0:00–2:00 in the weekends there was a user peek, which is possible that most of the contributions came from European countries.

To discover the reasons and identify our conjecture, we further analyzed user peeks during these two time slots. As shown in Fig. 5, from 4:00–6:00 most of the data was contributed from the U.S. and 10:00–12:00 European peers contributed the most (60–65%) but US peers still contributed over 20%. It indicates that intercontinental links (between the U.S. and Europe) are often used regardless of the number of local peers. For the third slot 0:00–2:00, peers from the U.S., Europe and others shared the similar portion of the contributions.

As network operators struggle to control the overall usage of bandwidth, it would be advisable that P2P service provider could constrain the media data transmission within a locality without frequent use of intercontinental links [15]. Otherwise, the inter-continental bandwidth usage will become a non-marginal issue

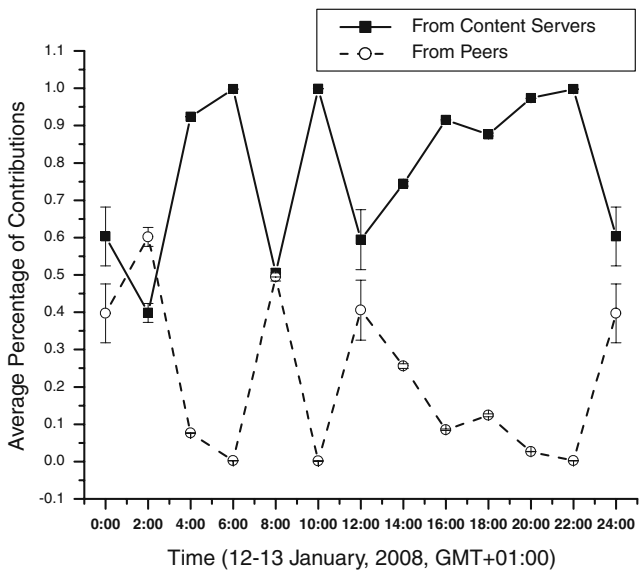

Fig. 4 Weekend trace of NAT/FW-behind node

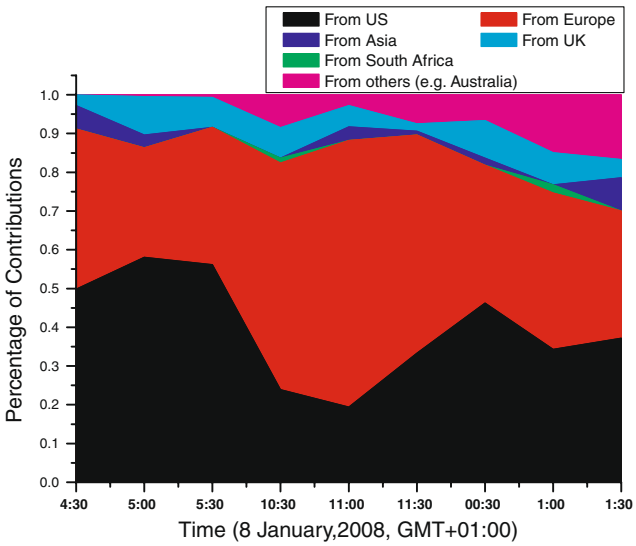

Fig. 5 Timeslot trace of NAT/FW-behind node

for the network providers. To verify whether locality-awareness has been carefully considered in Joost system, we studied its performance additionally in Section 4.3.5.

4.3.2 Time pattern—public node

Since public nodes and N/F-behind nodes ran the same channel, the available contributions were assumed to be similar. Surprisingly, public IP address configured nodes relied great heavily on the Joost content servers. As depicted in Figs. 6 and 7, most of the time content servers contributed over 60% of the media data. Comparing with Figs. 3 and 4, we conjecture that Joost uses a peer management algorithm similar to the one used in Skype that easily reachable nodes (e.g. public nodes) with high capacity are used to relay traffic for other peers, so-called super nodes in Skype. Actually, the difference of upload throughput between public nodes and N/F-behind nodes is significant (see Section 4.3.3). Besides, we believe that the available bandwidth, performance (e.g. CPU, memory size) are considered in the peer selection phase, which has been identified in [18].

If we consider the deviation of the weekly trace in Figs. 6 and 7, except for the time during 4:00–6:00, the rest hourly trace followed the similar pattern. To reveal the cause of difference, we tracked this particular time duration. It was noticed that on average 26.4% (total 49.3%) of contributions came from the US (local time: 20:00–1:00), 17.5% from Europe and rest of peers (e.g.

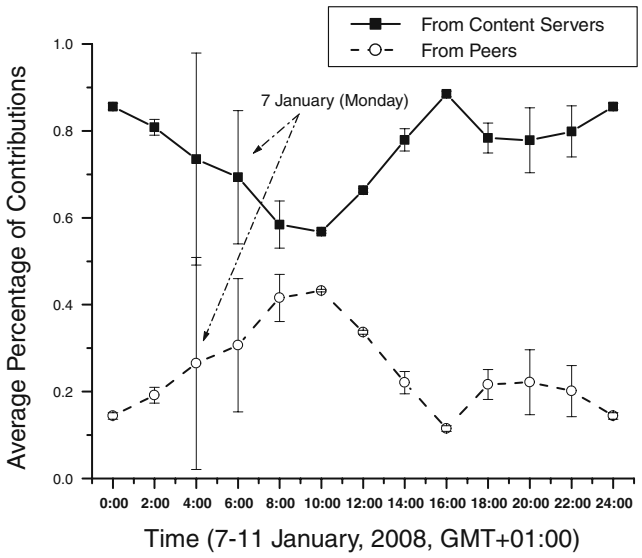

Fig. 6 Weekday trace of public node

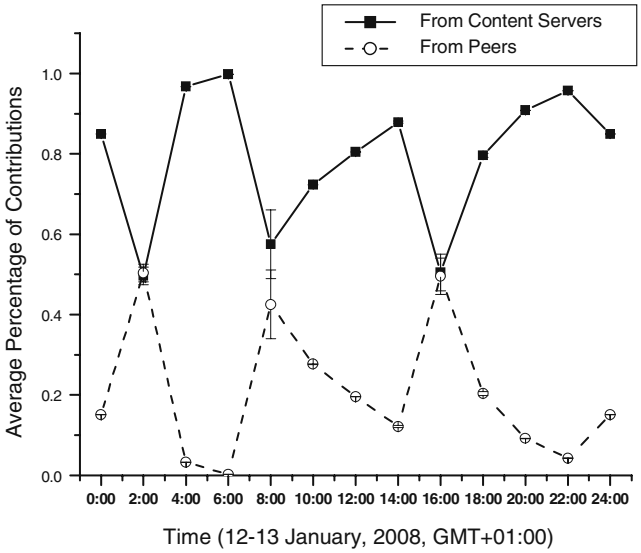

Fig. 7 Weekend trace of public node

Australia (GMT+10:00)) contributed 5.43%. Again, it conforms to the above observation that most contribution were transmitted through inter-continental links.

Furthermore, the weekend trace performed quite differently from weekday trace. Particularly, the time slots of 2:00 and 17:00 respectively reached the peer contribution peeks (over 50% of the contributions). Among these contributions, European peers transmitted on average 34.72% of the media data, US peers forwarded 11.0% and others contributed 6.48%. Nevertheless, Joost servers still contributed 47.8% which is much higher than any of the above contributions.

Lastly, the number of contributing peers is expected to be even higher at weekend nights but for both public and N/F-behind nodes most of the contributions came from content servers. As observed in [18], the possible explanation is that the local Joost users in Germany were limited likely due to the current programs are mostly only in English without German subtitles. If the Joost client can select preferred language in the secondary audio tracks, it will attract more clients during their relaxing time. The other explanation could be that most Internet TV fans are night owls since for both public node and N/F-behind node between 0:00 and 2:00 on weekends the peer contributions reached the highest peak. Another possible reason is that most German resident users were slow-speed ADSL/DSL users (e.g. 2 Mbps download and 1 Mbps upload bandwidth) and thus, their contributions might not be sufficient to support the high-quality playback requirement at other users.

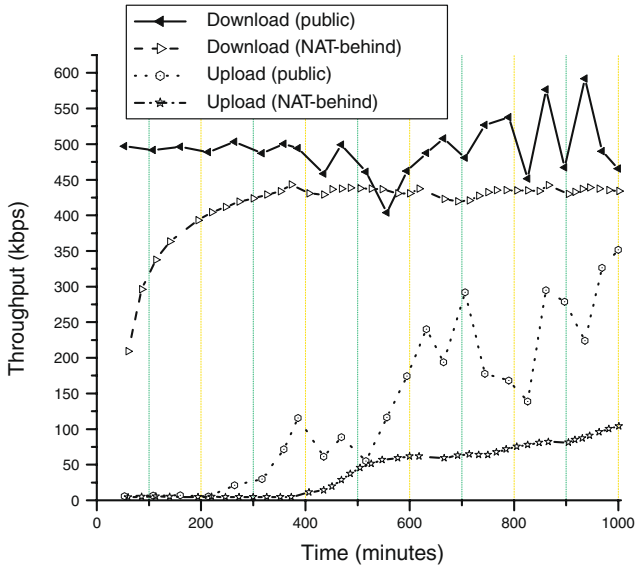

Fig. 8 Data throughput of public and NAT/FW-behind node

4.3.3 Bandwidth consumption

Having isolated the UDP packets, we examined the average throughput of public node and N/F-behind node for each period of 1000 minutes through four weeks. Figure 8 shows that the public node's upload throughput is on average 67% higher than that of the N/F-behind node although they have the same capacity regarding its bandwidth support and processing power. Such an observation suggests that public node is likely chosen as relaying nodes for other peers. Moreover, the average download throughput of the public node is 15% higher than that of the N/F-behind node since most of the data was directly transmitted from content servers. The reason why the throughput of the public node was not stable is that the content servers may not be able to contribute with the same amount of media data when simultaneously serving a large amount of peers.

We observed that the N/F-behind node downloaded and uploaded media data in a fairly constant speed compared with the public node. The download throughput of the N/F-behind node was 438 kbps, that is, 200 MB per hour, and 22 MB per hour for uploading. For public nodes, the average download throughput was 493 kbps, namely, 225 MB for one hour, and 68 MB per hour for uploading (3 times more than that of N/F-behind nodes). Through the experiments, we found that the average percentage of control traffic among the total traffic was 15%. Thus, public nodes only need to support 580 kbps downlink capacity and 84 kbps for uplink although they were connected to 100 Mbps full-duplex university LAN. Thus, we suggest that Joost can be a little aggressive especially to high-capacity node when they are available.

Further, we explicitly investigated low capacity nodes with ADSL connections in [18]. Unfortunately, these clients could be weakly (e.g. occasionally stalled) supported in the Joost system since current Joost system only provides the same quality for any video. We would suggest using layered or adaptive mechanisms for more efficient video distribution. For example, servers or some high-capacity nodes are responsible for transmitting the basic layer of the encoded video and other available peers could be used to transmit the enhanced layers in order to improve the video quality. Although it might introduce some complexities into the peer management, the Joost system could support much more users including some low-capacity nodes without wasting network resources.

4.3.4 Popularity impacts

As identified that pubic nodes actively participate in relaying media data for other peers, we only traced two public nodes running different types of programs as defined in Scenario 2.

As with many P2P media streaming applications, the number of users is largely determined by the popularity of selected program. For popular channels, the upload throughput should be much higher than that of unpopular channels. What is observed in Fig. 9 is consistent with the speculation that popular channel node's upload throughput was much higher (over 150%) than that of unpopular channel node.

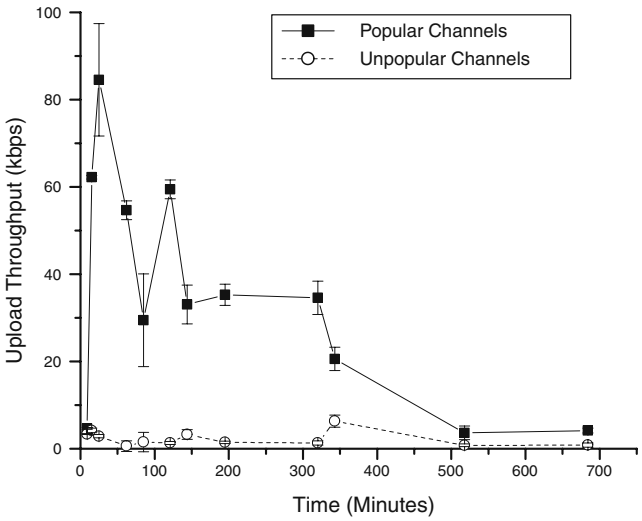

Fig. 9 Upload throughput of popular channels vs. unpopular channels

For popular channels, at the initial phase the throughput increased dramatically to reach the throughput peek 85 kbps. Then, it decreased till 35 kbps and increased again till 60 kbps and kept relatively low. The near constant upload throughput during the late stage suggests that Joost P2P system scales well since more contributors are able to forward media data after they receive the data.

During our experiments, the throughput of unpopular channels is always low (in average 2.34 kbps) and the unpopular channel node comes into the stable phase much earlier than the popular channel node. We conjecture that there are much less requests in the system for the unpopular programs.

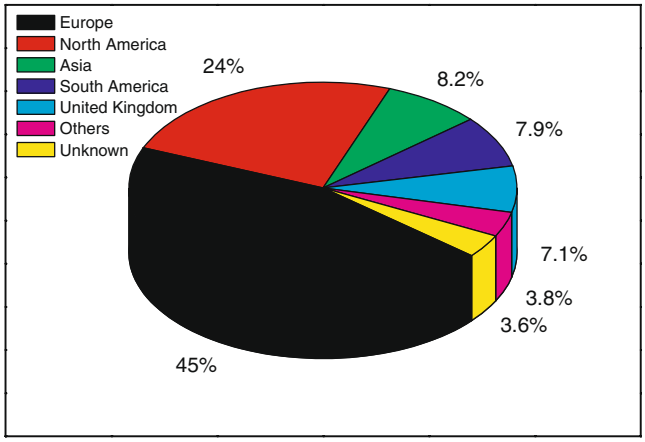

Fig. 10 Geographic location

4.3.5 Locality considerations

After three-day repeated experiments of Scenario 3, we analyzed the collected data from both test nodes. Although the two test nodes were watching the same channel and geographically locating near each other, the second test node only received in average 1.3% of the data from the first node (96.2 MB out of approximately 7.4 GB). Note that there are basically three different levels of locality: 1) geographical (e.g. same continent area); 2) AS-level (e.g. same AS); and 3) topological (e.g. same access network). The results can identify that the topological locality have not considered in the peer management. Otherwise, the second test node would receive most of the data from the first node instead from other remote nodes.

To further verify above observations, we parsed the IP address of contributed peers from which our test nodes received data. Totally, we identified 1210 distinct peers which provided inneglectable contents to our test nodes. These peers were located in over 54 countries. As shown in Fig. 10, the major sources of peers are Europe and the United States. Of all the data collected from the test nodes, 45% (547) came from European countries, 24% (293) came from the United States, 8.2% (99) came from Asian countries, 7.9% from South America, 3.6% from other countries. Besides, 45 IP addresses were not traceable and therefore we marked them as "unknown".

Moreover, sources of JCs from Germany were 130 (19% of Europe). Since our host was located in Germany, we suspect that the geographical distance (e.g. from specific continent) may have been considered in Joost. For example, the prefix awareness may have been considered during the peer selection. Note that a high-level (geographical locality) is not enough for peer management since resource can be still wasted for being transferred to a remote user in the same geographical location.

In summary, we ascertain that the locality-awareness is not well designed in the Joost system and can be improved with more considerations on topological level awareness.

5 Related work

Most of existing work about P2P VoD systems was concentrated on the protocol design and the implementation [8, 9, 12]. Other existing experimental analysis is based on Enterprise-supported data and focusing on translating daily "log" file into user arrival and departure distribution. Besides, the video data provided by other P2P media streaming systems [3, 10] is not seriously encrypted and the corresponding architecture is much easier.

Hei et al. [11] provided a comprehensive overview of P2P streaming system and characterized P2P IPTV behavior and traffic profiles at packet, connection and application levels. Their observations provided us a starting point towards comprehensive understanding of peer management mechanisms in Joost. Nevertheless, through experiments we identified the Joost architecture which is quite different from, even more complex than, any media streaming architecture described in [11]. The VoD functionalities give the users more flexibilities, and hence make the system more difficult to design as well as its reliability is more difficult to support. With regard to peer management, each Joost client is more "selfish" since it only cares about the content behind its current playback position.

Huang et al. [13] presented a general architecture for supporting P2P VoD services based on PPLive [3]. The

paper focused on introducing some important building blocks of PPLive system as well as evaluating user satisfaction and health of the systems. However, their focus is apparently different from ours, for instance, their measurement analysis does not consider popularity impacts and locality considerations.

Hall et al. [24] performed a measurement study of Joost in May, 2007. The authors showed a preliminary understanding of Joost's application behavior and network behavior. However, there are several major differences between their work and our work. First, their experiments were taken based on Joost version 0.9.2 which is now out-of-date. For instance, since January 2008 Joost changed the design to clean up the local cache after rebooting the client. Exactly because of that, the peer management can be very different from previous experiments. Through our extensive experiments we inferred the Joost architecture and key components. Moreover, we designed some typical scenarios in order to investigated the performance of peer management in terms of time pattern, bandwidth consumption and locality considerations, which was not provided in [24].

6 Conclusions and future work

Joost is one of the first commercial P2P VoD systems which can provide high quality on-demand TV based on P2P technologies. Unlike live media streaming system, each VoD client is more "selfish" in the sense that it only cares about contents after its current playing position, which is often different from other peers. The peer can only download from those whose playback positions are ahead, or from who have already downloaded the program. Instead, itself can help peers which join later than itself. However, as each client can change its playback position at any time, which differs from many other P2P streaming systems, it becomes difficult to optimize the overall VoD system. For example, the "rarest-first" strategy [7] in BitTorrent is not applicable here.

In this paper we have made a first step towards discovering various aspects of the Joost functions and behavior by analyzing the network traffic and by being acquainted with some of the open software used in Joost. We resort to data-driven analysis upon passive measurements to real the peer management mechanisms used in Joost. Our analysis demonstrates that with some dedicated infrastructure the current Internet infrastructure is capable of meeting performance requirements of high-quality VoD. Although large-scale P2P VoD systems are potentially deployed in today's Internet, the performance could be improved, for example, based on the following observations:

- The current Joost system relies on an overlay network deployed with a set of centralized content servers as identified in Sections 4.3.1 and 4.3.2, which may still raise a scalability issue in the near future. However, it is very useful to separate media distribution from controlling peer-to-peer hierarchy, which makes the Joost system relatively stable and potentially scalable.
- As indicated in Sections 4.3.2 and 4.3.3, we believe that public nodes with high capacity may be selected as main relaying nodes to ease the traversal of symmetric NATs and firewalls. However, in our experiments their uplink capacity usage is still quite low (on average 84 kbps). Thus, the Joost system could be aggressive to these nodes, together with certain incentive mechanisms to encourage them contribute more to the network, which may help the system overcome the scalability issue.
- Section 4.3.5 identified that the geographical distance may have been considered in the peer management of Joost. However, the lower-level locality-awareness (e.g. topological locality) may still be missing in the peer management. Besides, the inter-continental links are often used to transmit media data regardless of the number of local users, which may overload the network provider's costs. If the P2P service could be AS-/network level locality awareness, it would be beneficial for both customers and service providers.
- Joost currently provides each client with the same quality of video. This may result in an inefficient resource utilization if some clients are unable to support the desired video quality. Hence, layered video or adaptive mechanisms could be introduced into Joost.

As our experiments were conducted through passive measurements and data-driven analysis, above observations could be potentially limited in some perspectives. For example, the selected test-bed is relatively small due to the time and experimental limitations. Although we believe the major findings would be consistent with the fact behind the Joost system, it would be favorable to establish a large-scale testbed. Further, we mainly performed our measurements in Germany; however, it would be more interesting to conduct experiments in different locations (e.g. other European and Asian countries).

Besides, there are several avenues left for future work. For example, the exact peer lookup and selection techniques are still unclear. Our guess is that it uses a

combination of swarm mechanisms in BitTorrent and prefix awareness. Therefore, we intend to take a close investigation on Joost VoD functionalities, though difficult, which could provide more useful hints on how to provide efficient peer management methods.

References

1. Joost (2009) Joost homepage. http://www.joost.net/
2. scaryideas.com (2009) Joost network architecture. http://www.scaryideas.com/video/2362/
3. PPLive (2009). PPLive homepage. http://www.pplive.com
4. WildPackets (2009) WildPackets homepage. http://www.wildpackets.com/
5. Wireshark (2009) Wireshark homepage. http://www.wireshark.org
6. Abad CL, Yurcik W, Campbell RH (2004) A survey and comparison of end-system overlay multicast solutions suitable for network-centric warfare. In: Battlespace digitization and network-centric systems IV, pp 215–226
7. Cohen B (2003) Incentives build robustness in bittorent. In: 1st workshop on the economics of peer-2-peer systems, Berkley, CA
8. Do T, Hua K, Tantaoui M (2004) P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment. In: Proceedings of ICC'04
9. Guo Y, Suh K, Kurose J, Towsley D (2007) P2cast: peer-to-peer patching for video on demand service. Multimedia Tools Appl 33(2):109–129
10. Yu H, Zheng D, Zhao BY, Zheng W (2006) Understanding user behavior in large-scale video-on-demand systems. In: Proceedings of EuroSys'06
11. Hei X, Liang C, Liu Y, Ross K (2007) A measurement study of a large-scale p2p iptv system. In: The 6th international workshop on peer-to-peer systems
12. Huang C, Li J, Ross K (2007) Peer-assisted vod: making internet video distribution cheap. In: The 6th international workshop on peer-to-peer systems
13. Huang Y, Fu T, Chiu D-M, Huang C (2008) Challenges, design and analysis of a large-scale p2p vod system. In: Proceedings of ACM SIGCOMM
14. Norros I, Prabhu BJ, Reittu H (2006) Flash crowd in a file sharing system based on random encounters. In: Workshop on interdisciplinary systems approach in performance evaluation and design of computer and communication systems
15. IETF (2009) IETF application-layer traffic optimization (ALTO). http://www.ietf.org/html.charters/alto-charter.html
16. Almeida JM, Eager DL, Frris M, Vernon MK (2002) Provisioning content distrition networks for streaming media. In: Proceedings of INFOCOM
17. Kazaa (2009) Kazaa homepage. http://www.kazaa.com/
18. Lei J, Shi L, Fu X (2007) An experimental analysis of joost peer-to-peer vod service. Technical report no. IFI-TB-2007-03, Institute for Computer Science, University of Goettingen, Germany. ISSN 1611-1044
19. Level 3 Communications (2009) Level 3 communications homepage. http://www.level3.com/
20. Paul R (2006) More universities banning Skype. http://arstechnica.com/news.ars/post/20060924-7824.html
21. Baset SA, Schulzrinne HG (2006) An analysis of the Skype peer-to-peer internet telephony protocol. In: Proceedings of INFOCOM, Barcelona, Spain
22. Hilt V, Rimac I, Tomsu M, Gurbani V, Marocco E (2008) A survey on research on the application-layer traffic optimization (ALTO) problem. Internet draft (draft-ietf-alto-survey-00), ALTO
23. Jufsoft (2009) WhereIsIp. http://www.jufsoft.com/whereisip/
24. Hall YJ, Piemonte P, Weyant M (2007) Joost: a measurement study. Carnegie Mellon University, Pittsburgh

Jun Lei (lei@cs.uni-goettingen.de) received her Dr. rer. nat in Computer Science from the University of Goettingen, Germany in July 2008, and the M.E. degree in Engineering of Science from Zhejiang University, China, in 2004. She is also awarded with "Buchpreis der Fakultaet fuer Mathematik und Informatik". She is currently a Post-doctoral research fellow in the Institute of Computer Science, University of Goettingen, Germany. Her research interests include overlay networks, peer-to-peer networking, multimedia systems and mobile networks. She has served on a number of conferences as TPC member, including GLOBCOM, WiMob, SIMUTools, IWCMC and ICCCN.

Lei Shi (shi@cs.uni-goettingen.de) received his B.S. degree from Chongqing University, China, and his M.S. degree from Uppsala University, Sweden, both in Computer Science. He is currently a Ph.D. student at University of Goettingen. His research interests include network processors, P2P networks and TCP enhancements.

Xiaoming Fu (fu@cs.uni-goettingen.de) received his PhD degree in Computer Science from Tsinghua University, Beijing, China in 2000. After working at Technical University Berlin as a research staff for almost 2 years, he joined the University of Goettingen in 2002 as Assistant Professor, leading a research team working on networking research. Since April 2007 he has been a Professor and Head of the Computer Networks Group at the University Goettingen. His research interests lie in the architectures, protocols and mechanisms for QoS, firewalls, multicast, peer-to-peer, overlay, mobile networking and security related issues. He has published about 50 referred papers in international conferences and journals, as well as several RFCs/I-Ds. He has also served on a number of conferences as session chair, program chair or TPC member, including INFOCOM, ICDCS, ICNP, GLOBECOM, ICC, MobiArch. He chaired the 1st and 2nd ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch) and currently serves on its steering committee. He is a member of editorial board of Elsevier's Computer Communications Journal, and a guest editor of the IEEE Network's forthcoming Special Issue on Implications and Control of Middleboxes in the Internet.