Peer-to-Peer Netw Appl (2009) 2:75–83
DOI 10.1007/s12083-008-0015-5

Characterization of signaling and traffic in Joost

Majed Alhaisoni · Antonio Liotta

Received: 17 April 2008 / Accepted: 7 October 2008 / Published online: 18 December 2008

Abstract Peer-to-Peer (P2P) IPTV applications have increasingly been considered as a potential approach to online broadcasting. Recently, many applications such as PPlive, PPStream, and Sopcast have been deployed to deliver live streaming via P2P. One of the latest systems is Joost, which can deliver both Video-on-Demand and Real-Time services. Measuring and characterizing this application in terms of signaling overheads and traffic profiles helps to better understand the key limitations of current P2P IPTV systems. Therefore, the main purpose of this paper is firstly to study the impact of Joost on the network. Secondly, we wish to determine the underlying mechanisms of Joost, distinguishing between the Video-on-Demand and the Real-time services. Our study is carried out through a close investigation and analysis on the traffic of Joost in two types of streaming. Based upon the data tracing and collection, many different statistics have been derived. Our study unveils strengths (e.g. good resilience to end-to-end delay and jitter) and shortcomings (e.g. poor locality) and yields recommendations for future P2P IPTV systems.

Keywords P2P streaming · Traffic characterisation · Joost

M. Alhaisoni
University of Essex,
Wivenhoe Park,
Colchester, CO4 3SQ, UK
e-mail: malhai@essex.ac.uk

A. Liotta (✉)
Department of Electrical Engineering,
Eindhoven University of Technology,
Den Dolech 2, P.O. Box 513, PT 11.29,
5600 MB Eindhoven, The Netherlands
e-mail: A.Liotta@tue.nl

1 Introduction

Peer-to-Peer (P2P) streaming represents an economical, robust, and scalable alternative to the more conventional client–server (CS) approach [11]. The basic idea is that, rather than streaming media from dedicated servers, an application-level overlay is formed by the user terminals, which cooperate in the distribution of the stream itself [11]. While receiving a stream, terminals simultaneously act as distribution hubs for it. In this way, the bottlenecks and failure points traditionally associated with servers are virtually eliminated, since individual servers are replaced by a multitude of user terminals. Also, as the number of users connecting to a certain stream grows, so is the number of distribution points. Hence the system scales much better than any CS counterpart [11].

The P2P streaming concept has now lead to a number of trial P2P IPTV systems such as PPlive [1], Joost [2], PPStream [3], and Sopcast [4]. There is now clear commercial interest in these new technologies which are revolutionizing the online broadcasting arena.

Despite the numerous advantages of P2P streaming in general and P2P IPTV in particular, their characteristics in terms of signaling overheads and network efficiency are not well known. Most systems make use of proprietary protocols and are not open to the research community. This makes it virtually impossible to carry out simulation studies aimed at determining whether a large-scale deployment of P2P streaming will be sustainable in terms of network resources, operation, and management.

On the other hand, trial platforms are already in use, which creates the opportunity to get a better insight into mechanisms of P2P and their effects on the network. Our work pursues this avenue. We look at signaling and traffic characteristics in Joost, one of the most recent applications

supporting both Video-on-Demand (VoD) and Real-Time (RT) streaming services. Data tracing and deep packet inspection unveiled a wealth of interesting properties, some of which were not entirely expected. Our study reveals strength (e.g. good resilience to end-to-end delay and jitter) and shortcomings (e.g. poor locality) and yields recommendations for future P2P IPTV systems.

2 P2P streaming architectures

Peer to Peer streaming architectures can be categorized based upon their distribution mechanisms. The various approaches to P2P streaming have been surveyed by Liu et al. [11]. Our analysis is particularly concerned with two main methods, which are briefly outlined below.

2.1 Tree-based architecture

In the tree-based architecture, the peers are ordered hierarchically by the source, known as the parent. The parent node, in turn, sends data packets to intermediate nodes, and these nodes relay them iteratively until leaf nodes are reached.

Despite introducing a good level of parallelism and distribution, this approach suffers from a number of limitations. The root, or data source, is a single point of failure, which limits the robustness of the system. Another problem is that if peers join and leave frequently the tree has to be rebuilt too often, which has a negative impact on signaling overheads, latency, and stability.

An example of tree-based streaming application is Peercast [12], an open-source software for streaming both audio and video. A peculiarity of Peercast is that any node can specify the maximum number of incoming connections allowed.

2.2 Mesh-based architecture

In this architecture, the overlay network supporting the stream distribution is a mesh. Data is divided in chucks in such a way which allows a peer to receive portions of the stream from different peers and assemble them locally.

This approach is more robust than the tree-based architecture, since when a stream comes from various sources communication does not break when only a subset of peers disconnect. Another benefit is that this transport method reflects well the asynchronous nature of many access network technologies (e.g. ADSL). In fact, peers can download a stream at full quality whilst uploading only a fraction of it. The same feature is exploited in P2P file sharing application BitTorrent [14]. Common streaming applications in the mesh-based category are GNuStream [13], PPlive [5], Coolstreaming [9], and Joost [2]. This method is becoming more widespread than the tree-based one. That is why in our study we have chosen to scrutinize Joost.

3 Related work

Very few researchers have studied P2P streaming based on commercial platforms. Applications that can be truly categorized as P2P have appeared only recently and their underlying algorithms are, in many cases, proprietary and not readily available for academic scrutiny.

A study which is similar to the one presented in this article has been carried out by Hei et al. [1, 6]. They characterize the traffic and behavior of PPlive, whereas we have focused on Joost which is more recent and provides both VoD and RT streaming. Other studies focused only on the VoD service are reported in [7, 8], and [9].

Given to its original conception, Joost has several similarities to Skype, which is however a P2P conferencing tool. From the several works assessing Skype we can learn about mechanisms that are supposedly analogous in Joost. Examples include the bootstrapping process, the supernodes' election algorithm, and the management of the overlay when nodes join or depart [10].

At the time of writing, only two studies of Joost are available in the form of technical reports. Lei et al. explain the key underlying mechanisms of Joost [11]. Lei et al. focus on Joost components and architecture [5]. These studies were conducted on earlier versions of Joost (v. 0.9.2 and v. 1.0, respectively). Our study is based on v.1.1.4 which is more stable and includes new functionality, most notably RT streaming. Because of this, we have been able to identify key differences between VoD and RT, in addition to carrying out further statistics on packet traces. Also we characterize both UDP and TCP traffic, both in upload and download modes.

Our work looks specifically at network locality, the ability of Joost to select nodes which are close to each other. We characterize the relative distribution of nodes and the mapping between logical and physical overlays at city and country levels.

Tobias Hoßfeld et al. [15], have done a comparative study of popular P2P IPTV systems such as Joost, PPlive, and Zattoo. Their study was concerned about how the user perceives these P2P IPTV systems. In addition, they have indicated key features of these applications, including topology management, distribution protocols, and bandwidth utilization. Similarly to our study, they conducted experiments based on passive measurements and considered both download and upload traffic.

Our study considers additional parameters, studies both RT and VoD streaming and looks specifically at network locality and geographic load distribution.

Another study of P2P streaming system, focused on Coolstreaming, is by Susu Xie et al. [16]. They used real traces and tried to draw some theoretical basis to demonstrate that selecting peers randomly has the possibility to scale well. They developed some fundamental concepts about Coolstreaming, showing how problems relating to heterogeneity can be addressed by some advanced buffering techniques. They also looked at important performance factors including bootstrap time and the rate of join and leave in the P2P network.

Similarly to Xie et al., we study P2P IPTV in terms of churn and bootstrapping delay. However, we also look at other parameters such as traffic distribution and network locality. We considered upload and download traffic incurred by RT and VoD. Most importantly, our work can be considered complimentary to Xie's since we draw conclusions based on real measurements, rather than relying on simulations.

Another work that focuses on the issue of network locality is by Aleksandra Kovacevic et al. [17]. They conclude that location awareness decreases transmission delay, one of the most important factors in media streaming. Despite working on the assessment of different systems, our study leads to similar conclusions about the importance of designing location-aware overlays to pursue bandwidth conservation.

Another eminent work is by Thomas Silverston et al. [18] who have performed a measurement study during the last FIFA World Cup, comparing four P2P streaming applications (PPStream, PPlive, TVAnts, and Sopcast). Their study was concerned with the traffic statistics and the churn of peers. In addition, they were concerned about the impact of the traffic generated on the network from these applications. Finally, they showed the user behavior into these systems. Likewise, we present our results based on passive measurements of live streams. However, our work focuses on Joost which is a more recent application and supports both RT and VoD services. Additionally, we study network locality.

Alexandro Sentinelli et al. [19] report on their measurements based on Sopcast. They figured out a number of important parameters like the average number of peers that a node connects to, the typical start-up delay, the continuity indexing, and the amount of data buffered at the client side. Their work is particularly valuable as it is based on a large-scale testbed, PLANETLAB. Again, our work can be considered to be complementary as we look at additional features and study Joost.

4 Experimental method

4.1 Overview of Joost

Joost is a peer to peer streaming application that delivers television-quality VoD and RT streaming services via a P2P network overlay. It was created by the founders of Skype and KaZaA and is currently in Beta version (v. 1.1.4 at the time of writing) [5]. Joost supports more than 15,000 TV programs through more than 200 channels [2]. Key components and functions of Joost are described below.

At system start-up, Joost performs a number of operations:

- Port selection

During the initial bootstrapping, Joost selects a specific port to connect to and communicate with other peers via UDP. The Internet Assigned Numbers Authority has recently assigned port 4166 for this purpose.

- Local Cache

Each Joost Client stores all the media data as "anthill cache" in the C drive, using the following directory: C:\Documents and settings\Application Data\Joost\ "anthill cache". Anthill [20] is an agent based supporting the media distribution services. The size of the local cache depends on how and for how long the program has been launched, so it increases with the size of the programs. However, in case of watching the whole channel, all the media data has been stored in the local cache, so in the second watching, the channel will be running from the local cache instead of connecting to the server except for some Codec. In our experiment, it caches more than 2GB. However, this will affect the user resources when more channels are watched.

- Installation

One of the functions of Joost is the Installation, so in this phase the Joost client connects to the server sending an HTTP request to retrieve the available channel list and download the SQLite file [21] which gives the initial available channel lists. Moreover, SQLite is used for managing the database of Joost channels.

- Bootstrapping

In Joost, there are three servers and two super nodes. Initially, the JC (Joost Client) connects to the server lux-www-lo-2.joost.net over HTTP. Then, JC will receive some available super nodes. After that, an HTTP request will be sent to lux-www-lo4.joost.net to get the updated version of the software. Lastly, the JC will connect to the super nodes such as lid-snode-1-eth0.joost.net to get the available list of channels and the available peers that are watching the channels. Before that, JC has already started communicating with other servers and peers. A schematic view of Joost's architecture is depicted in Fig. 1.

4.2 Experimental setup

Our experiments were conducted in the United Kingdom. We collected Joost packets in different types of streaming

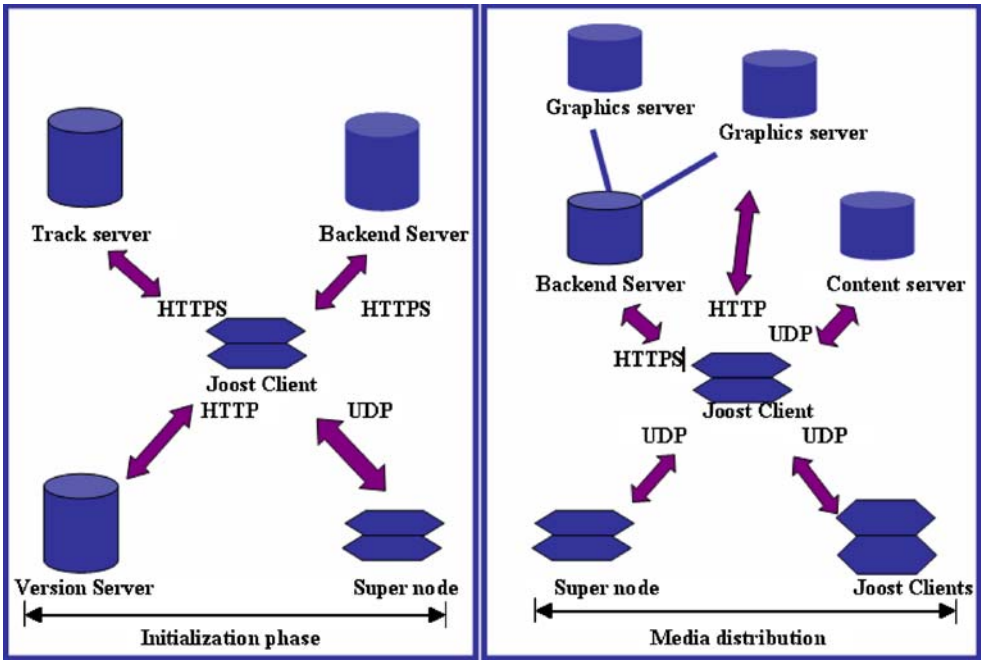

Fig. 1 Joost architecture [5]

mode (VoD and RT). We used the current beta version of Joost, v. 1.1.4.

Figure 2, outlines the set up of the two machines used for the collection of traces, which were then used for packet analysis. The machines were connected to a 100 Mbps Ethernet, connected to the campus Internet leased line. This ensured that both inbound and outbound bandwidth were considerably higher that then minimum required for the correct functioning of Joost.

The specifications of the machine used to gather VoD data were as follows: Windows XP with Intel® core™ 2CPU 6420 @ 2.13 GHz, 3.25 GB of RAM.

The specifications of the machine used to gather live streaming data were as follows: Windows XP with AMD Athlon TM 64 3400+ processor (2.10 GHz), 1GB of RAM.

Each PC ran Wireshark v1.0.0 (an open source network protocol analyzer which is known as Ethereal) and Netpeeker v3.10, a network monitor used to capture all inbound and outbound traffic incurred by Joost.

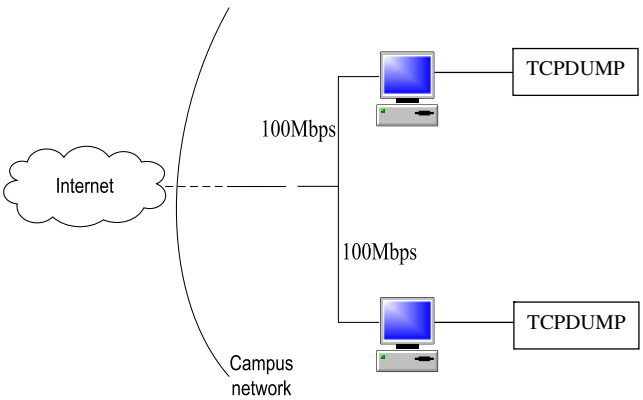

Fig. 2 Experimental environment

4.3 Data collection

VoD traces were collected from one of the most popular Joost channels *i.e.*, selected from the "what's popular" section. Traces included all events based on a 2-h observation window. The overall size of our trace files was 277 Mb.

RT traces were based on the four live channels which broadcast sport clips. Our traces include all events based on a 1-h observation window relating to a single live channel. Broadcasting is currently fairly limited to just a few hours per day, which limited the amount of data that could be collected within our experimental timeframe. The overall size of our trace files was 104 Mb.

Data traces were filtered at different granularities *i.e.*, at byte, packet, and session levels. We distinguished between UDP, TCP, upload, download, data (VoD and RT), and signaling packets.

Signaling packets were differentiated from data packets since the former are in the order of 60–70 kb, whereas the latter are between 800 and 1,000bytes.

5 Traffic characterization and underlying mechanisms

5.1 Start-up delay

Start-up delay is one of the important factors in online broadcasting. High values will give a low user experience,

since the user is used to very low start-ups associated to television sets.

In P2P streaming, a start-up delay is unavoidable. The time incurred between the request to connect to a channel and the actual start of playback, is mainly due to: (1) the underlying peer discovery mechanisms, while the P2P overlay selects suitable peers that can act as uploaders (or data sources); (2) buffering time, required to deal with network jitter, packet loss, and temporary congestion.

For VoD, we found an average start-up delay of 25 s, ranging from a minimum of 16 to a maximum of 35 s.

On the other hand, RT streaming incurred a start-up delay of 5s, ranging from a minimum of 4 to a maximum of 8 s. The reasons for this considerable difference between VoD and RT can be found by analyzing some of the data described later in this article. In essence, buffering times in RT are smaller than their VoD counterpart since the former is transmitted at a lower bit rate and gives higher priority to responsiveness rather than quality. Also the discovery time incurred whilst determining the sources will be lower, since RT connects to an average of three sources whereas VoD requires an average of five sources (Figs. 7 and 6, respectively).

5.2 UDP vs. TCP utilization

Joost uses both UDP and TCP, although our analysis shows a wider and more frequent use of the former. Super nodes periodically exchange small UDP and TCP signaling packets (64bytes) with other peers for overlay management purposes (*e.g.*, to check whether relevant simple peers are still reachable). Furthermore, every time a user switches channel, the peer needs to get in touch with its super node, which handles the stream re-direction process. However, our data shows that UDP packets are used more frequently for this purpose.

Data (video) packets are transported via UDP (of approximately 1 kb). The differences between UDP and TCP protocols utilization for the cases of VoD and RT are visible from Figs. 3 and 4.

The fact the RT packets are only observed for about 20 min is because this is currently the average length of live broadcasting session in Joost. Figure 3, shows that UDP incurs a higher rate at the beginning which is due to the need to buffer as fast as possible, a requirement which is less important in VoD. However, looking at the long-term average of transfer rate, we can see that RT is encoded at a lower rate than VoD. Clearly the extra constraints introduced by RT require relaxing some of the requirements on the quality of the stream.

Figure 4, gives an account of the different pattern observed in RT and VoD in terms of signalling. In fact, TCP is not used for data transfer in Joost. We can see the RT requires a higher peak of signalling at the beginning, but

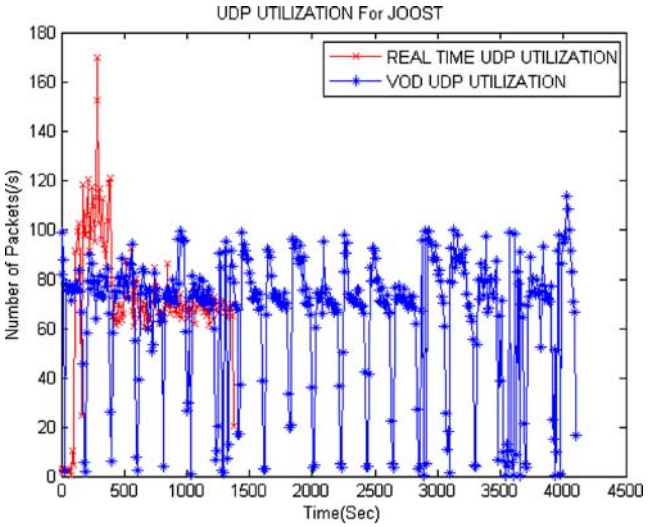

Fig. 3 UDP utilization

the long-term average goes below its VoD counterpart. We have not been able to define reasons for this. However, this behavior suggests that RT initiates a more aggressive search for sources, since it requires connecting quickly. And this is also in agreement with the lower connection times seen in RT (5 s as opposed to the 25 s of VoD), as observed in Section 5.1.

5.3 Download video traffic

In this section we evaluate the traffic coming into a peer (download mode). Throughput, as depicted in Fig. 5, is measured by adding all video components (or sub-streams) which constitute an individual session (either VoD or RT). Bit rate is obtained by averaging over a 10-s window.

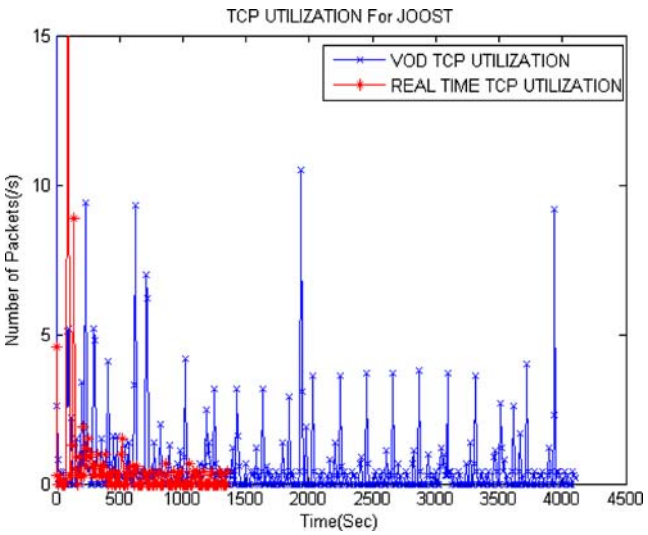

Fig. 4 TCP utilization

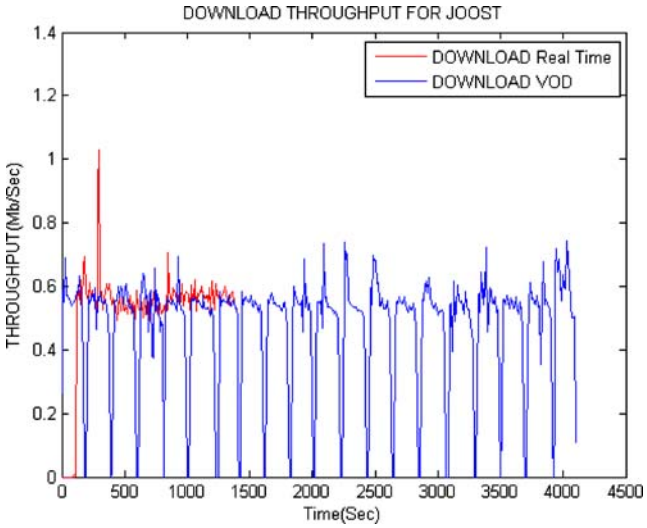

Fig. 5 Download throughput

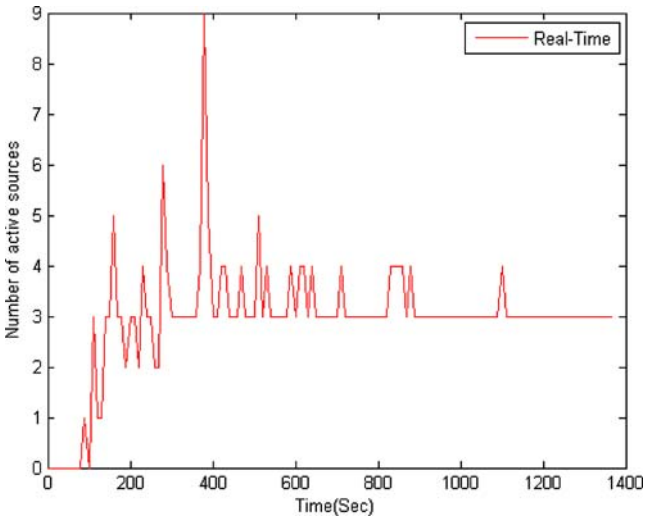

Fig. 7 Number of active sources (RT)

The most notable result from Fig. 5 is that RT is encoded (thus transmitted) at a lower bit rate than VoD. We have already commented on this aspect before in Section 5.2, however, this behavior suggests that RT initiates a more aggressive search for sources, since it requires connecting quickly. And this is also in agreement with the lower connection times seen in RT (5 s as opposed to the 25 s of VoD), as observed in Section 5.1.

We also observe that the variability in VoD transmission rate is much higher and several intervals do not see any packet transmission at all. In VoD this is acceptable since playback deadlines are less stringent than in the case of RT.

A greater insight is achieved by looking at Figs. 6 and 7. The number of sources is on average 5 in VoD (Fig. 6), by contrast to the average of 3 in RT (Fig. 7). We also studied how frequently sources changed during the transmission and found confirmation that this is considerably higher in VoD (data not included for brevity). This means that the VoD overlay management algorithm prioritizes load balancing (this results from the frequent connections and disconnections between peers, and especially in VoD for finding good quality of transmission), whereas RT prioritizes a smoother transmission. So once a set of sources is identified, RT tries not to change them during transmission, to prevent loss of quality or transmission intermittency.

5.4 Upload video traffic

Upload traffic is incurred when other peers connect to the peer under scrutiny (in our testbed). This data is very useful in establishing how Joost handles the asymmetry typical of many access networks (*e.g.*, ADSL allows for higher download rates than upload ones). Figure 8, shows that this requirement is, in fact, satisfied by Joost. Both RT

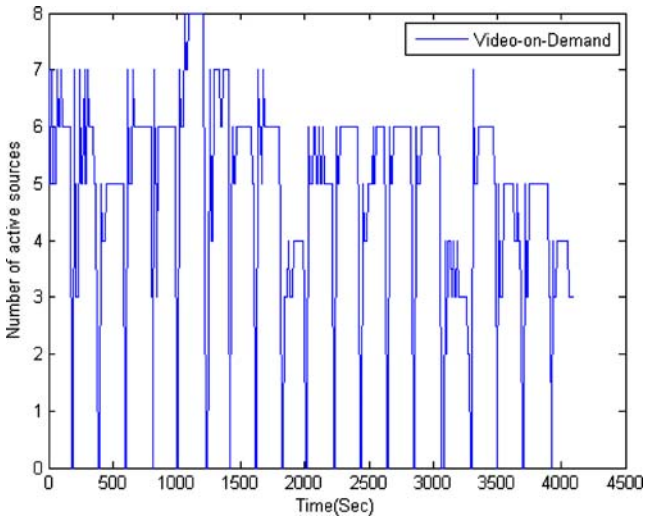

Fig. 6 Number of active sources (VoD)

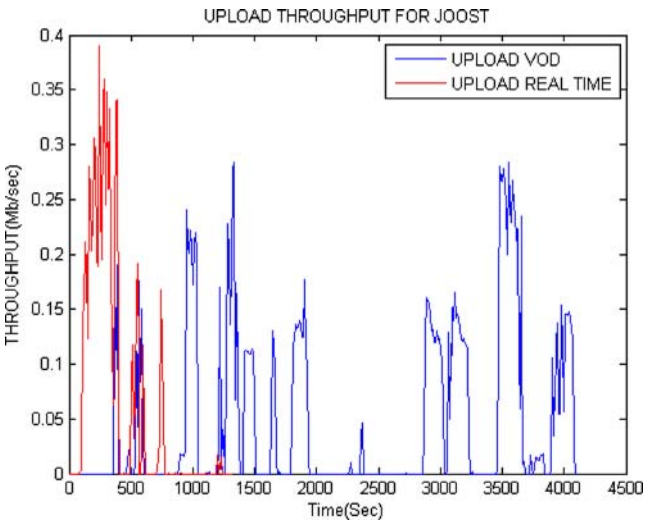

Fig. 8 Upload throughput

and VoD figures are considerably lower than their download counterparts. The average rate for the uploading for the RT was 0.68 Mbps and for VoD was 0.46 Mbps.

Consistently with Fig. 8 are our observations of inbound connections. Joost is configured in a way as to allow on average 1 inbound connection, both in RT and VoD. The pattern of connectivity is, however, different. RT inbound connections are mainly concentrated in the initial part of the overall broadcast period but are relatively more stable and incur a higher rate than in VoD. This is coherent with the need for greater stability in RT, where handover among different sources is kept to the minimum.

On the other hand, VoD gets inbound connection request during a longer period, although there are longer periods having no connections. This reflects the continuous handover among alternative peers for load balancing purposes (as already noted above).

Worth mentioning is that when we studied the correlation between inbound and outbound connections among peers, we noticed that there is no reciprocity. That is to say that, contrary to many P2P file sharing systems such as BitTorrent, Joost does not employ a tit-for-tat policy. In other words, nodes do not try to act as uploaders for those peers who have previously provided data.

5.5 P2P streaming applications taxonomy

Table 1 shows how Joost performs and works compared to other popular P2P streaming applications with respect to different parameters.

6 Network locality and geographic distribution

Network locality is the ability to maintain the P2P overlay in such a way as to create logical connections among peers who are physically close to each other. The way intercommunicating peers are geographically distributed has significant implications in terms of network efficiency—the ideal condition being when the most intensive data exchanges happen among nearby peers.

In order to determine whether Joost is location aware, we have conducted a number of experiments, monitoring the IP addresses of all communicating peers. We then used IPNETinfo v1.10 to map IP addresses to physical (geographical) addresses—at city and country levels. As before, we considered VoD, RT, upload, and download traffic. However, while in the experiments described above we were studying traffic and connections over the time; in the following pie-charts we display the average geographical distribution of inbound and outbound connections. A selection of results is presented below.

Table 1 P2P streaming applications

P2P application	Service	Start-up delay	Data rate	Architecture	Protocol
Sopcast	Live/ VOD	1–	5 min	300– 350 kbps	Mesh UDP
PPlive	Live/ VOD	20 s–	2 min	500 kbps	Mesh TCP
Joost	VOD	25 s	500 kbps	Mesh	UDP
Zattoo	Live	6.2 s	560 kbps	Tree/ Forest	TCP

6.1 Download traffic

Looking at Figs. 9 and 10, it is clear that Joost does not provide a good implementation of the concept of network locality. In fact, connections with our UK-based test bed are scattered across the globe. It is ironic that only a small fraction of data comes from the UK (26%). If we then breakdown the distribution of this UK inbound traffic, most of it originates from cities that are farther away.

Unexpectedly, most of the VoD traffic originates from Canada whereas most of the RT traffic comes from the USA. This seems to infer that Joost load balances computing resources but neglects network resource optimization.

6.2 Upload traffic

The analysis of upload traffic allows studying the distribution of peers connecting to (streaming from) our test bed (Figs. 11 and 12). Again, we notice a network-unfriendly behavior. However, neither VoD nor RT traffic go to those countries that were providing the highest percentage of download data, that is Canada for VoD (Fig. 9) and the USA for RT (Fig. 10). Also the portion of outbound traffic going to the UK (11%) was all directed to a unique location (Watford), which is far away from our test-bed site (Fig. 12).

These results confirm our previous comment about the intrinsic unfairness of Joost which does not implement a tit-for-tat policy. It is therefore hard to draw any definite conclusion as to whether Joost implements an explicit computational load balancing algorithm. It is possible that a high degree of randomness is embedded in overlay management.

7 Concluding remarks

In this paper we have conducted an experimental study on one of the most recent P2P streaming applications, which features genuine P2P mechanisms and supports both RT

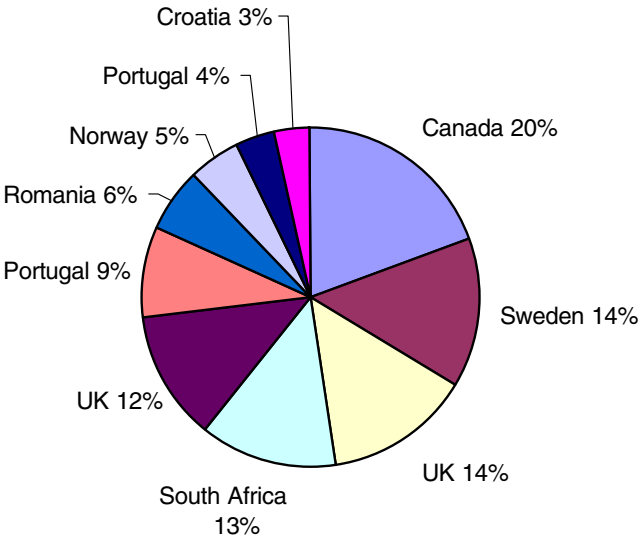

Fig. 9 Download VoD

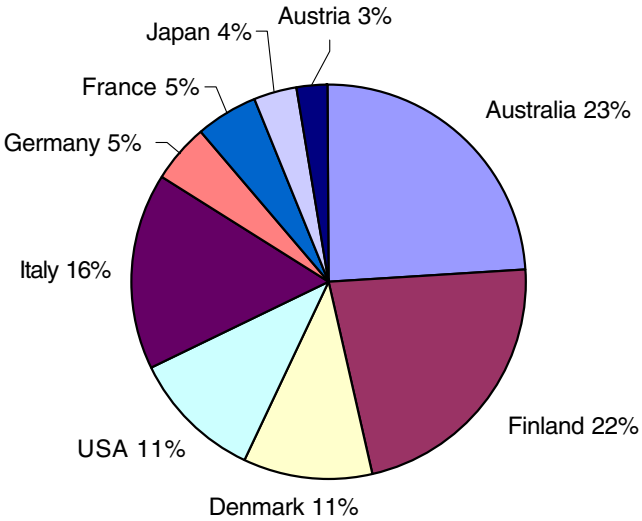

Fig. 11 Upload (VoD)

and VoD services. Through passive measurements on those services we have characterized some underlying mechanisms, including protocol behavior and signaling overheads. This has lead to the identification of important factors and issues that are common to the more general area of P2P broadcasting, P2P IPTV (including VoD and RT services), and P2P conferencing. Lessons learned from this study, which can yield interesting further developments and investigations, can be summarized as follows:

- Load balancing: P2P systems are known for their ability to autonomically balance computing resources. This is achieved well in P2P file sharing applications where time constraints are not so stringent, which allows for better resource optimization and overlay management. However, P2P streaming poses time limits which make this task harder. Methods which cater for near-optimal load balancing of computing resources within the limits imposed by RT streaming will acquire importance since mobile, thin terminals will demand more effective offloading mechanisms.
- Network locality: streaming without considering ways for optimizing the use of network resources is bound to pose serious hurdles, since this clashes with the *modus operandi* of network operators and ISPs. Methods which allow prioritizing connections based on geographical proximity as well as mobility patterns have considerable potential in terms of network efficiency.
- Fairness and free riding: this is one of the issues in common to all type of P2P applications, not merely P2P streaming. There must be a way to prioritize connections based on mutual resource sharing. There is, on the other hand, a strong trend towards free riding (*i.e.*, peer who get resources without sharing their own), a problem which dramatically degrades the performance of P2P. In the case of P2P streaming, it seems easier to tackle this issue. Appropriate mechanisms, coupled with economic models (*e.g.*, based on incentives) will be required.

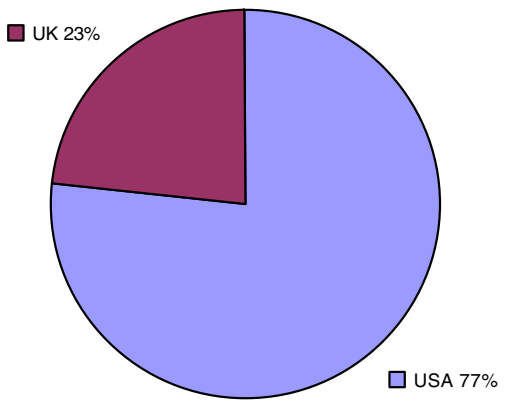

Fig. 10 Download (RT)

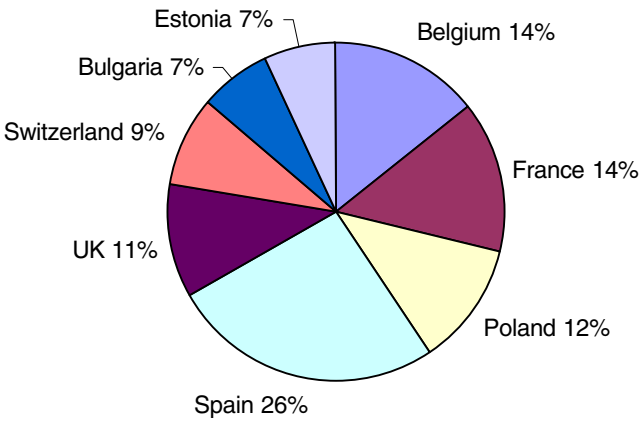

Fig. 12 Upload (RT)

- Start-up delay: this is excessive in current systems, which makes the user experience poor if compared to conventional TV broadcasting. Switching channels is also too slow, which makes zapping impossible. This seems a hard problem since buffering time cannot be reduced in current best-effort networks.
- Mobile user: current P2P streaming systems impose requirements (in terms of computational and access network capability) that are beyond the reach of state-of-the-art mobile phones or PDAs. Mobile P2P streaming poses some challenging research issues that are bound to attract vivid attention.

References

1. PPlive, www.pplive.com. Last Access was on 17th of April 2008
2. Joost, www.Joost.com. Last Access was on 18th of April 2008
3. PPStream, www.ppstream.com. Last Access was on February 2008
4. Sopcast, www.sopcast.com. Last Access was on January 2008
5. Lei J et al (2007) An experimental analysis of Joost peer to peer VOD services. Technical Report, Computer Networks Group, University of Gottingen, Germany
6. Hei X, Liang C, Liu Y, Ross K (2007) A measurement study of a large scale P2P IPTV system. IEEE Trans Multimed 9(8):1672–1687
7. Do T, Hua K, Tantaoui M (2004) P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment. Proc IEEE ICC 3:1467–1472
8. 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
9. Huang C, Li J, Ross K (2007) Peer-assisted VoD: making internet video distribution cheap. In Proc. IPTPS 2007, Feb 26–27
10. Zhang X, Liu J, Li B, Yum T-SP (2005) DONet/CoolStreaming: a data-driven overlay network for peer-to-peer live media streaming. Proc IEEE INFOCOM 3:2102–2111
11. Liu Y, Guo Y, Liang C (2008) A survey on peer-to-peer video streaming systems. Peer-to-Peer Netw Appl 1(1):18–28 March
12. Diot C, Levine BN, Lyles B, Kassem H, Balensiefen D (2000) Deployment issues for the IP multicast service and architecture. IEEE Netw 14(1):78–88
13. Jiang X, Dong Y, Xu D, Bhargava B (2003) GnuStream A P2P media streaming system prototype. Proc 2003 ICME 2:325–328
14. Qiu D, Srikant R (2004) Modeling and performance analysis of BitTorrent-like peer-to-peer networks. SIGCOMM 34(4):367–378
15. Hoßfeld T et al (2008) A qualitative measurement survey of popular internet-based IPTV systems. Appears in the International Conference on Communication and Electronics, 4–6 June 2008, pp 156–161
16. Xie S et al (2007) Coolstreaming: design, theory, and practice. IEEE Trans Multimed 9(8):1661–1671
17. Kovacevic A et al (2008) Location awareness—improving distributed multimedia communication. Proc IEEE 96(1):131–142
18. Silverston T, Fourmaux O (2007) P2P IPTV measurement: a comparison study. Submitted on 23 Oct 2006 (v1), last revised 19 Apr 2007 (v4)
19. Sentinelli A et al (2007) Will IPTV ride the peer-to-peer stream. IEEE Comm Mag 45(6):86–92
20. Babaoglu O, Meling H, Montresor A (2002) Anthill: a framework for the development of agent-based peer-to-peer systems. In Proc. 22nd IEEE International Conference on Distributed Computing Systems 2002, Vienna, Austria
21. Hipp R. SQLite. http://www.sqlite.org/

Majed Alhaisoni earned his BSc in Computer Science from Qassim University, Saudi Arabia (2005) and his MSc in Computer and Information Networks from the University of Essex, UK (2007). He is now a PhD candidate at the Department of Computing and Electronic Systems, University of Essex, UK. He research interests encompass heterogeneous wireless systems, peer-to-peer networking and applications, and distributed computing. His PhD is in the area of P2P streaming.

Dr Antonio Liotta is a Professor of Communication Networks at the Eindhoven University of Technology in The Netherlands. He is a Fellow of the U.K. Higher Education Academy and serves a number of senior advisory boards including: the Peer Review College of EPSRC (the UK Engineering and Physical Sciences Research Council); the scientific advisory panel of the IWT (the Belgian Research Council); the Board of Editors of the Journal of Network and System Management (Springer); and the advisory board of editors of the International Journal of Network Management (Wiley). He is an active member of the networking research community. He has co-organized and co-chaired international conferences; has served the Technical Programme Committee of over 70 conferences; and has also contributed as keynote and tutorial speaker. At the University of Essex, Dr Liotta was leading the Pervasive Services team, known for its pioneering work on ubiquitous computing, advanced service management, and systems engineering. He a prolific and enthusiastic writer, with over 100 scientific publications to his credit in the areas of telecommunication services, distributed computing, and advanced networking. Recent articles have contributed to topical themes including: operator-mediated peer-to-peer; mobile grids; quality of experience management in context-aware services; and the legal issues of pervasive systems.