Joost: A Measurement Study

Yensy James Hall, Patrick Piemonte, and Matt Weyant
School of Computer Science
Carnegie Mellon University

May 14, 2007

Abstract

The potential to broadcast on-demand television-quality video to a global audience is now possible with the introduction of peer-to-peer Internet television. Joost, originally known as The Venice Project, is a peer-to-peer technology created by the founders of Skype and KaZaA to deliver television-quality, licensed video content. It had taken researchers quite some time to understand the mechanisms and technologies used by Skype, so this paper will begin to unravel the mysteries of Joost through a measurement study. Our contribution will be an understanding of Joost's application behavior, network behavior, and peer behavior. We will also attempt to explore reasons of peer selection as well as uncovering the benefit of Joost using peer-to-peer as a method of content distribution. Our hope is to provide this information in order to assist in future design or modeling of peer-to-peer video distribution systems.

1 Introduction

Originally known as The Venice Project, Joost is a peer-to-peer application that could revolutionize the future of entertainment media distribution. It was created by the founders of Skype and KaZaA to deliver television-quality video on-demand over a peer-to-peer network. This paper will describe our results of a measurement study to shed light on Joost's application, network, and peer behavior with intent to assist the future design and modeling of peer-to-peer video distribution systems.

1.1 Resources

We conducted our study using three distinct Joost user accounts on three separate test machines. Each test machine–or test node–was configured to operate in a specific network environment.

The first test node ran the Mac Intel beta build of Joost version 0.9.2 while connected to a residential Comcast cable-modem with 3Mbps of downlink capacity and 1Mbps of uplink capacity. This node was installed behind a network address translator (NAT) and configured with a non-routeable, private IP address. The test node also had an Intel Core 2 Duo processor, 1 GB SDRAM, and a 120 GB hard drive. The second and third test nodes ran the Windows XP beta build of Joost version 0.9.2 while connected to the Internet from the same subnet on the Carnegie Mellon University network. These machines were connected to the same switch, and were granted 100Mbps full-duplex connections. Each of these university test nodes had a Pentium 4 processor, 2GB of RAM, a 320GB SATA hard drive, and 100Mbps Ethernet adapters. These machines were also given static, public IP addresses.

1.2 Data Collection

Using these test nodes we constructed a data collection infrastructure to capture traces of Joost activity. A number of software packages were installed to facilitate our data collection. The two test nodes connected to the 100Mbps full-duplex university LAN were running Windows XP Service Pack 2 with Wireshark, a packet capture tool based on the WinPcap

3.1 library. The cable-modem test node was running Mac OS 10.4.9 along with tcpdump and the latest version of the pcap library.

Joost was run on all machines, with the packet capturing software, for a combined total of nineteen days, non-continuously, between the dates of April 10th, 2007 and May 3rd, 2007. Altogether, the data collection infrastructure captured a total of 64 GB of data.

1.3 Experiments

Using the captured data, we began our analysis by approaching various questions we had regarding Joost application behavior and network behavior. These questions are explored by category in the following list.

1.3.1 Application Behavior Experiments

Our initial experiments were intended to provide information on the basic application functionality as well as to determine initialization and reconnection protocols. We examined the captured packets for the use of encryption and plain text clues, such as HTTP header information. Analysis was then performed on connection sequence packets. Addresses from these packets as well as various web sources would yield Joost infrastructure information. We then took a black box approach to these experiments and blocked the various ports at which Joost operates in order to find functionality changes. As a last application behavior experiment, we analyzed packet sizes for control packet and video data packet ratios.

1.3.2 Network Behavior Experiments

Next, we characterized Joost's network behavior. We measured and compared the inbound and outbound throughput of Joost on both the cable-modem connection as well as the machines connected to the university LAN. We also measured how much data was transferred across the university's switch to assess the local impact of collocated Joost peers.

1.3.3 Peer Behavior Experiments

Finally, we examined locality awareness with the following three experiments.

1. We used two test computers (let's call them A and B) connected to the 100Mbps full-duplex university LAN and physically located next to each other. Both computers were always synchronized on the same Joost channel and one of them (A) was always about 1 minute ahead. The idea was to see if the computer that was a little behind (B) would get its data from the other one (A). With this setting, we collected 9GB of data, non-continuously, during a four-day period.

2. We collected over 5GB of data continuously over a 23 hour period in one of our test machines connected to the university LAN. We then extracted the distinct IP addresses of Joost peers that transmitted data to our client, and for each address we found its geographical location by sending an HTTP request to "http://api.hostip.info/get_html.php?ip=" passing the IP address as a parameter and parsing the response. The response was given in plaintext, e.g "Country: UNITED STATES (US) City: Pittsburgh, PA". With this approach, we could not find the location of 31% of the IP addresses which contributed 4.58% of the data.

3. We collected 22.1GB of data during a 13 days period in one of our test computers connected to the university LAN. These data was not captured continuously. Our goal was to accumulate enough data to identify patterns in the selection of peers over several days.

2 Application Behavior

By analyzing the Joost client application and by reading information from its website, we concluded that the application is based on XULRunner, a Mozilla runtime package [11]. Joost also decodes video using the CoreAVC H.264 codec developed by

CoreCodec [8]. With this technology, Joost creates a P2P overlay that distributes video content.

Like its sibling P2P networks, KaZaA and Skype, the signaling traffic between peers and the Joost servers is encrypted. However, by examining packet traces we can determine the connections that a Joost peer makes when it connects to the P2P network. Unlike the work in [2], Joost does not include a specific HTTP header to distinguish its traffic from the rest of the normal Internet traffic. Therefore, we needed to observe how Joost initially and recurrently joins the network.

2.1 Installation and Bootstrapping

First, the peer initiates a secure HTTPS connection to www.theveniceproject.com. Then, the peer initiates three more encrypted HTTPS sessions. Each of these encrypted sessions connects to lux-backend.joost.net; however, taking a closer look at the client hello for each of these sessions shows the actual server being requested: tolbiac.ops.theveniceproject.com, adengine.ops.theveniceproject.com, and tracker.ops.theveniceproject.com respectively. Luckily, two of these servers have descriptive domain names leading us to believe that adengine.ops.theveniceproject.com serves the video advertisements that play between episodes and that tracker.ops.theveniceproject.com plays a similar role to that of a BitTorrent tracker. Then, the peer checks the current version of the software through an HTTP GET request to version.ops.theveniceproject.com. Finally, the peer connects to a supernode (snode-1.lid.ops.theveniceproject.com, in this case) to establish connections with other peers and begin accessing video content.

At some point during this initialization process, the peer negotiates a port number through which communication with other peers and supernodes occurs. Although the exact protocol for negotiating this port was not identified, it is highly likely that a modified STUN protocol was used, similar to what [4] and [5] observed in their analyses of Skype. Moreover, the Joost client uses this port number for every subsequent video transaction. The consistent usage of this port enabled us to observe and measure video transfers and transactions.

2.2 Reconnecting

When reconnecting to the Joost network, the same initialization process occurs. However, the port number that was negotiated when first connecting to the network is cached and reused. Next, the peer attempts to reconnect to peers from which it has downloaded content previously. The peer sends a small UDP probe (∼30 bytes) to each peer, which in turn respond with another small UDP packet (∼30 bytes).

2.3 Joost Server Infrastructure

Joost distributes licensed video content, and unlike many P2P networks, user-generated content is not permitted. Thus, all new content is introduced through its own servers. Joost is headquartered in Europe, and as a result, much of its infrastructure is also in Europe. In fact, the company maintains servers in data centers in Belgium (212.8.163.0/24), the Netherlands (89.251.0.0/23 and 213.207.101.128/25), and the United Kingdom (212.187.185.0/24). However, much of the content licensed is also meant for audiences in the United States, so Joost also maintains a farm of servers in Los Angeles, California (4.71.105.0/24).

Our analysis of packet traces indicates that Joost has engineered portions of their infrastructure for specific tasks. Servers in the 89.251.0.0/24 subnet are administrative servers and supernodes. Supernodes in this subnet transmit relatively little data, which indicates that they are likely serving channel lists and updated peer lists. Several servers in this subnet are also the main European web servers for joost.com.

Servers in the 212.8.163.0/24 subnet include the tolbiac, adengine, and tracker servers mentioned during session initialization. Also, a vast amount of video traffic was received from servers in this subnet, leading us to believe that the primary content servers also reside in this subnet.

In fact, during the course of our study we observed the introduction of two new data centers: Los Angeles [9] and London [10].

2.4 Controlling Joost

Using our residential test node on the Comcast high-speed data connection, we conducted several experiments to learn how Joost can be controlled within a network's boundary.

Historically, controlling P2P traffic within a network and at its borders is difficult. Firewall traversal protocols like STUN and TURN are used by P2P applications to gain traction within network boundaries. Thus, predicting on which ports two peers will communicate and then blocking traffic on those ports is nearly impossible. P2P applications also do not confine themselves to a single transport layer protocol, using UDP and TCP with equal frequency. Finally, the distributed nature of P2P makes it nearly impossible for a network administrator to determine whether an IP address is a P2P node or a legitimate user. Additionally, P2P networks have become open markets for distributing copyrighted video without permission; so, P2P applications make it intentionally difficult for network administrators to stem the flow of P2P traffic.

But because Joost distributes licensed video content, it does not need to obfuscate its transport mechanisms in the same way as Gnutella or other P2P networks, making the network security engineer's job much easier. We conducted several experiments to learn how Joost can be controlled within a network's boundary. The results of these experiments can be found in Table 1.

From these results, several conclusions can be drawn. First, blocking TCP port 80 or 443 is an effective means of preventing Joost from bootstrapping the P2P network. However, wholesale denial of service for these ports also prevents legitimate use of the web. Second, Joost nodes appear to negotiate a port number on installation using a STUN-like protocol; but instead of renegotiating this port number on reconnect, the local Joost application retains the originally negotiated port number. Moreover, if this port is blocked, the application will not attempt to negotiate an open port. Therefore, a security administrator could attempt to block every local port number of each Joost node within his network, but this is an intractable problem. Moreover, blocking this port does not stem the flow of traffic from the Joost-operated content nodes. Third, ordinary Joost nodes receive video from Joost-operated content servers that operate on port 33333. Thus, traffic from Joost-operated nodes can be blocked, but this does not prevent traffic from other ordinary Joost nodes from entering the network.

In summary, network operators are unlikely to be able stop users from installing Joost P2P clients within the network boundary. Because the Joost client uses common HTTP and HTTPS ports to communicate with supernodes and administrative servers, denying Joost its signalling channel is also unlikely. Blocking individual port numbers is an intractable solution. Thus, the best strategy left is to block port 33333 at the network border[1]. Doing so kills the Joost client's playback capability, and because the majority of video traffic comes from Joost-operated nodes (see Section 4.1.2), we have limited the amount of data flowing into and out of the network significantly.

2.5 Control and Data Traffic

We analyzed the raw data to determine the breakdown control traffic versus video traffic. Using a technique from [2], we used packet sizes to determine the relationship between control and data traffic.

Figure 1 shows the distribution of packet sizes for all of our collected data. We compared the distribution of packet sizes with and without TCP ACK packets, but the graphs were almost identical. We can conclude that at least 45% of the packets are part of control and acknowledgment traffic, and the remaining 55% is video traffic. Packets under 200 bytes are considered control packets, and data packets are 1000 bytes or more.

[1]While this paper was being written, Joost applied for and received approval to use TCP and UDP port number 4166. This port will be used in the Now For Friends edition of the client software as well as subsequent versions.[12]

Experiment	**Results**
Block 80/tcp during installation	Installation succeeded; bootstrap failed
Block 80/tcp during video playback	Playback failed
Block 443/tcp during installation	Installation succeeded; bootstrap failed
Block 443/tcp during video playback	Playback failed
Block 33333/udp and 33333/tcp during video playback	Playback failed; data from ordinary nodes continued
Block 45587/udp and 45587/tcp during video playback	Playback failed; data from Joost-operated nodes continued
Block 8.251.0.0 during installation	Installation succeeded; bootstrap failed
Block 212.8.163.0 during playback	Playback failed

Table 1: Results of Joost security experiments

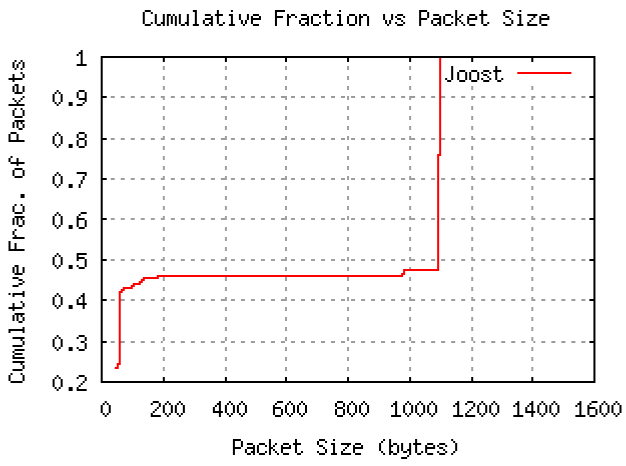


Figure 1: CDF of packet sizes

3 Network Behavior

We conducted an analysis of our data for information on how Joost's inbound and outbound throughput affects a network. In particular, we would like to know the amount of bandwidth it consumes and how local network traffic could be affected from a large adoption of Joost users.

3.1 Rate

Our measurement results show that Joost uses approximately 700 kbps down and 100 kbps up, regardless of the type of network the node is residing on (Figure 2). Whether a node is on a high-speed LAN with a static public IP address or on a lower-speed broadband cable-modem with a NATed IP address, we found that the throughput of Joost remains consistent while the peer is online.

We realized this throughput behavior may be constrained on the cable-modem connection in Figure 2 (a), so we monitored the university LAN test nodes for any variation. The university LAN test nodes used slightly more uplink capacity when it was available, but it was still a minimal amount. Figure 2 (b) shows the slight increase to about 150 kbps. This indicates that Joost could be slightly more aggressive with a larger uplink capacity when it is available.

3.2 Local Traffic

In terms of Joost adoption and local network traffic, our results could be subject to the fact that Joost is not prone to increased peer sharing since the majority of the content is pulled from the Joost infrastructure nodes. In section 4, we examined the relationship between same switch peers and local impact more closely.

4 Peer Behavior

4.1 Locality

4.1.1 Geographic Locality

From experiment 1 in section 1.3.3 we found that computer B received only 1.03% of the data from computer A (0.062 GB out of approximately 6 GB). Taking into account that the vast majority of the data that B needed was already in A, we can conclude that Joost is missing an opportunity to use network resources efficiently.

In experiment 2, section 1.3.3, we identified 6918 distinct peers providing content to our test computer during the 23-hour period. These peers were located in approximately 980 cities in 90 countries. Of all the incoming data collected, 34.1% came from unknown cities in the United States, 32.37% came from unknown cities in the United Kingdom, and and 20.76% came from unknown cities in Europe.

As Figure 3 shows, the main sources of peers are the United States and Europe. We did not see any other area providing a substantial number of peers.

To determine the correlation between distance and amount of data transferred, we found the amount of data that each known city contributed, and manually estimated the distance between the city and Pittsburgh. For the unknown cities in the U.K. we chose the distance between Pittsburgh and London, while for the unknown cities in Europe, we chose the distance between Pittsburgh and Paris. To find the highest correlation possible, we assumed that the rest of the unknown locations were in Pittsburgh, so we assigned distance = 0 to all IPs from an unknown country or from an unknown city in the US. These IPs provided 39% of the data.

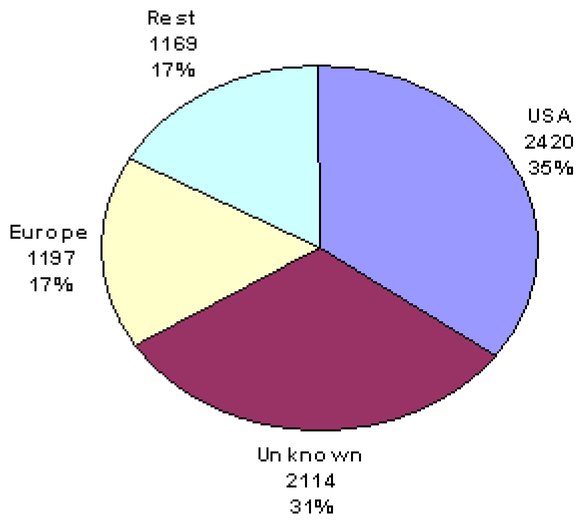
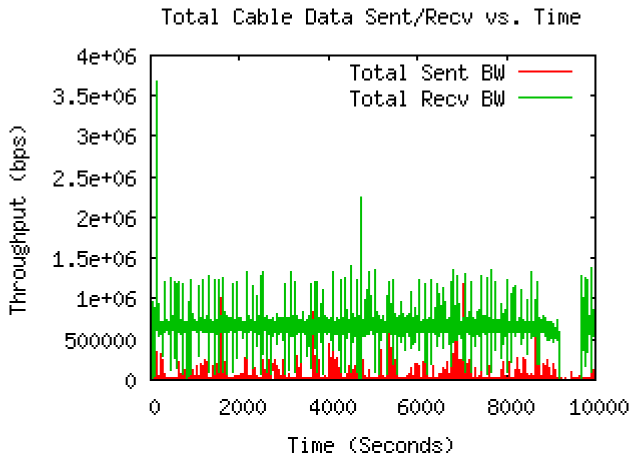


Figure 3: Breakdown of Joost traffic by geographic location

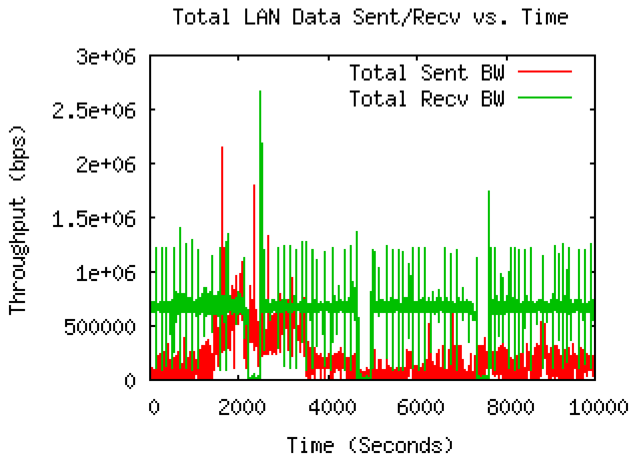
Even with the assumptions explained above, the correlation between geographical distance and amount of data was -0.013. In other words, we found no correlation. We did a similar analysis with the data received only from the United States, and the correlation was -0.054 (also no correlation).

Of the data collected in experiment 3, section 1.3.3, 14.4 GB was inbound traffic to our test computer. From that data, we identified four IP prefixes (4.71.105.0/24, 212.187.185.0/24, 212.8.163.0/24, 89.251.0.0/24) that provided a disproportionate amount of data, and found out that they belong to Joost. We then determined that these IP prefixes provided 71.24% of the incoming data.

The results of these three experiments indicate that Joost is not considering geographic locality in the selection of peers. Furthermore, Joost servers are directly providing users with at least two-thirds of the content.



(a) Residential cable-modem 3Mbps down/1Mbps up connection



(b) 100Mbps full-duplex university LAN connection

Figure 2: A comparison of throughput over time

4.1.2 Network Locality

In addition to geographic locality, we wanted to learn whether Joost took network proximity into account when selecting peers. We set up a Joost client connected to a 100Mbps full-duplex university local area network with a static IP (the "university" client) and a Joost client connected to a residential cable-modem connection with a NATed IP address (the "home" client). The residential connection had a downlink capacity of 3Mbps and an uplink capacity of 1Mbps. For this experiment, we used round-trip time (RTT) to determine whether two peers were close to each other. We observed the Joost clients during a one-hour video program, because this is more likely to be representative of user behavior than letting the clients run for many hours at a time. This user behavior is analogous to how many people watch television. We passively captured packet traces of each of the clients while viewing this program. After distilling the data to find unique transmitting nodes, we concentrated on the nodes that contributed 1MB or more to the video stream. We then measured the RTT from the receiving client to the transmitting node using scriptroute's ICMP and TCP ping facilities. In particular, we wanted to see if Joost's peer selection algorithm takes advantage of smaller RTT to reduce video latency and the cost of large amounts of data flowing across transit ASes.

During this experiment, we observed different behavior between the university and home clients. The home client had only one set of 9 peers that transmitted over 1MB of video, and all 9 peers in this set were Joost-owned content servers from the same /24 subnet located in the United States. The average RTT of these peers was 91 milliseconds. These nodes contributed 98% to the overall amount of data transferred for the hour-long program. The university client received video from 15 distinct peers, in addition to the Joost-owned peers, that contributed over 1MB to the video stream. Moreover, these independent nodes contributed 45% of the total data in the video stream, compared to 52% which is contributed by the Joost-owned content servers. The independent peers ranged in RTT from 25 to 103 milliseconds with a large cluster of nodes between 60 and 90 milliseconds. The Joost-owned content servers are 87, 91, and 115 milliseconds away for servers located in the UK, US, and the Netherlands respectively. A summary of this experiment can be found in Table 3.

Additionally, during the course of this experiment, Joost released a software update for its client application that performed a look-up to determine which Joost-owned content servers to use. The intent of the release was to increase performance for US-based Joost clients. However, because our clients were located close to the east coast, we received data from Joost-owned servers in California and London in nearly equal amounts.

For the university client, there is no mathematical correlation between the amount of data received and the RTT of the transmitting node. In fact, the smallest contributor in the top 15 has an RTT within 6 milliseconds of the mean (standard deviation = 22.01), and the most distant peer delivers the median amount of data. The home client received 98% of its data from nodes 91 milliseconds away, but these nodes represent less than 10% of the peers during the hour-long program (Figure 5(b)). The remainder of the home client's peers had substantially worse RTTs. However, it appears there may be some type of RTT-savvy selection algorithm at work. The peers that transmitted more than 1MB of video to our university client had RTTs that were comparable to the RTTs of Joost-owned content nodes. But independent nodes that transmitted data to the home client had significantly worse RTTs than the Joost-owned peers. Thus, we can infer that the peer selection algorithm may prefer to receive data from peers that have an RTT less than or equal to the RTT to the Joost content nodes.

Looking at Figures 5(a) and 5(b), it is interesting to note that roughly 80% of university client peers are within 200ms but only 10% of home client peers are within the same threshold. It is clear that the university client had many more peers nearby; thus, there may be a relationship between the selection of nearby peers and the upstream or downstream bandwidth available to our client.

Unfortunately, RTT may not be a reliable indicator of actual network proximity. Bottleneck links, link failures, route asymmetry or AS peering relationships may affect the RTT. We have left this portion of the analysis for future work.

4.1.3 Topological Locality

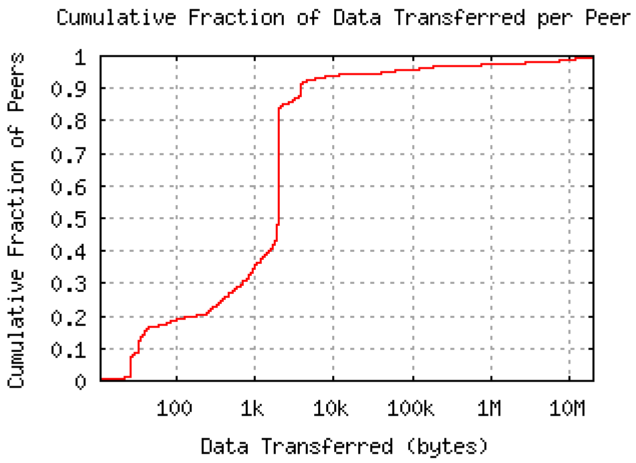
Round-trip time is only one measure of network proximity. We also considered topological locality using traceroute hop counts from our receiving node to the transmitting nodes. We conducted this experiment in parallel with the RTT locality experiment. We used scriptroute's ICMP and TCP traceroute facilities to determine the number of hops between our local Joost client and a transmitting peer. Specifically, we looked at the transmitting peers, who are not owned by Joost, that contributed more than 1MB of data. For a one-hour program, 15 peers contributed more than 1MB of data to the university client's video stream. Unfortunately, four nodes could not be reached. We used a web-based traceroute server to confirm that these nodes were not reachable via traceroute. Because the home client only received more than 1MB of data from Joost-owned servers on the same subnet (18 hops away), we will focus on the university client. Table 3 summarizes the results of this experiment.

Our client received 80.44MB of video data from the 11 peers listed in Table 3, which represents 31% of the total data received. The peers have hop counts ranging from 15 to 30, but the smallest contributor has a hop count near the mean and the median. In this case, "near" means well within one standard deviation. However, these hosts exhibit a weak negative correlation between hop count and the amount of data transferred. But this correlation contradicts our intuition that nodes topological closer to our client will transmit more data. Thus, it is unlikely that Joost selects peers based on topological locality.

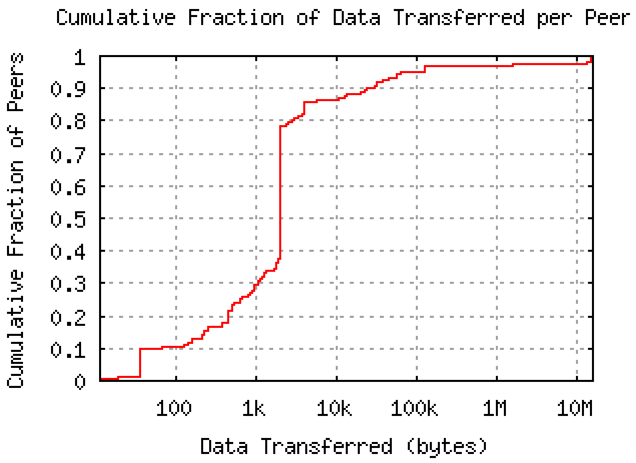
4.2 Fairness

In this paper, we define fairness of a peer as the ratio between the video bandwidth sent to the video bandwidth received by the peer. From the data collected in experiment 3 of section 1.3.3, we found the fairness ratio to be $7.7/14.4 \sim 1/2$. Also from the data collected in experiment 1 we obtained a fairness ratio of $9/20 \sim 1/2$.

These results make sense. Since Joost is directly providing at least two thirds of the video content to

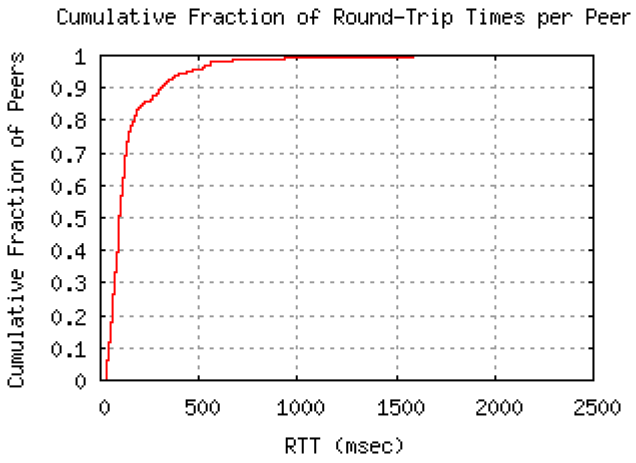


(a) University client. Joost client connected to a 100Mbps full-duplex LAN with static, public IP address.

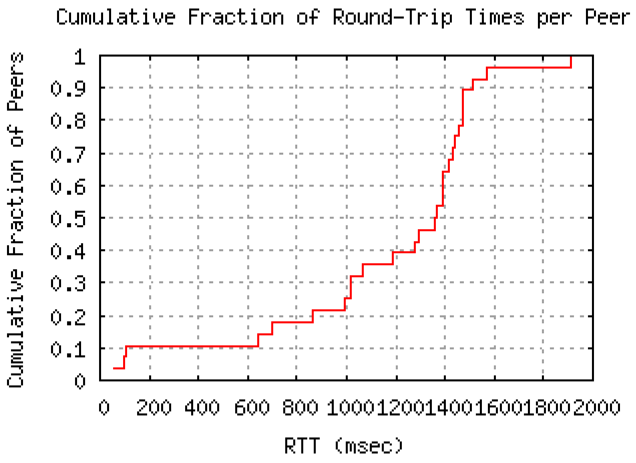


(b) Home client. Joost client connected to a 3Mbps down/1Mbps up residential cable-modem connection with a NATed IP address.

Figure 4: CDF of data transferred per inbound peer



(a) University client. Joost client connected to a 100Mbps full-duplex LAN with static, public IP address.



(b) Home client. Joost client connected to a 3Mbps down/1Mbps up residential cable-modem connection with a NATed IP address.

Figure 5: CDF of RTT per inbound peer

Server	**Location**	**RTT**	**Hop Count**
Joost admin server (89.251.4.175)	NL	113 ms	15
Joost version server (4.71.105.59)	US	91 ms	19
Joost supernode (89.251.0.16)	NL	113 ms	14
Joost supernode (4.71.105.10)	US	91 ms	17
Joost content node (212.8.163.11)	BE	115 ms	17
Joost version server (213.207.101.226)	NL	97 ms	16
Joost content server (212.187.185.143)	GB	87 ms	14

Table 2: Table of RTT and hop count from CMU LAN to a representative subset of Joost-operated nodes using scriptroute's ICMP and TCP ping and traceroute facilities.

Host Name	**RTT (msec)**	**Hops**	**Data (MB)**
Host 1	86	16	18.64
Host 2	68	17	18.55
Host 3	59	15	17.74
Host 7	89	17	8.06
Host 8	78	19	4.67
Host 9	103	30	3.80
Host 10	101	20	2.63
Host 12	25	15	2.27
Host 13	61	18	1.70
Host 14	69	15	1.36
Host 15	80	19	1.02
Mean	**74.45**	**18.36**	**7.31**
Median	**78**	**17**	**3.80**
Standard Deviation	**22.01**	**4.27**	**7.32**
Correlation to Data Transferred	**-0.002**	**-0.316**	**N/A**

Table 3: Summary of topological locality experiment for the university client.

its clients, only one third will have to be supplied by independent nodes. We think that Joost's low fairness ratio is sustainable now only because of the relatively low user population, and that it will be a problem in the future in terms of scalability. Hence, we anticipate that it will change to a number closer to 1 as the number of users increase.

5 Related Work

There have been many peer-to-peer measurement studies, but to our knowledge none have studied Joost. A comprehensive measurement study of KaZaA's overlay structure is undertaken in [3]. This paper provides pointers to several other studies that cover:

- The identification of peer-to-peer traffic
- Techniques to analyze and characterize peer-to-peer traffic and file-sharing workloads
- The development of crawling systems for Gnutella, Napster, and KaZaA
- A study of scalability issues in the Gnutella network
- A study of the problem of search and replication strategies in unstructured peer-to-peer networks
- A study about the advantages of designing unstructured peer-to-peer systems based on super peers

Joost was created by the founders of both KaZaA and Skype, which have both been studied extensively ([2], [3], [4], [5]). These studies provided some intuition on how Joost may have been engineered. In particular, [2] measured the workload of KaZaA and showed that a P2P file-sharing workload does not exhibit a Zipf distribution, which has become the common statistical model for Web traffic. This paper also showed that KaZaA users are patient, often waiting hours or days to receive the requested file. [2] also discovered that KaZaA does not take advantage of locality to manage its overlay network and proposed performance improvements if KaZaA were locality-aware. [4] and [5] examined Skype's application behavior and supernode infrastructure. [3] looked in depth at the KaZaA and FastTrack overlay networks with both active probing and passive trace analysis.

P2P video distribution systems have also been studied ([1], [6]). Reference [6] studied an on-demand P2P video system called GridCast. GridCast was deployed in China in 2006 to over twenty thousand users, hundreds of which were supported concurrently.

6 Conclusion

Although Joost is a peer-to-peer video distribution technology, it relies heavily on a few centralized servers to provide the licensed video content. We believe the centralized nature of Joost is the main factor that influences its lack of locality awareness and low fairness ratio. We see scalability issues in this approach and therefore predict a more distributed architecture in the future.

From a network usage perspective, Joost consumes approximately 700 kbps downstream and 120 kbps upstream, regardless of the total capacity of the network. This is assuming the network upstream capacity it is larger than 1Mbps.

Joost consistently uses port 33333 to send video data from Joost's infrastructure servers. This should allow network administrators to significantly limit the amount of data flowing into and out of the network by blocking port 33333 at the network border.

Although we noticed that there may be some type of RTT-savvy selection algorithm at work, we could not find any mathematical correlation between RTT and the amount of data transferred from a peer to our client.

7 Summary

To summarize, we performed a measurement study of Joost, a peer-to-peer licensed video distribution service. Our intent was of curiosity, to explore a new technology to determine its effect before it is publicly released. We analyzed 65GB of captured packet data from three test nodes over a combined total of nineteen days. One test node was used for experiments on a residential cable-modem connection with 3Mbps of downlink capacity and 1Mbps of uplink capacity. The other two test nodes were setup for experimentation on Carnegie Mellon University's 100Mbps full-duplex LAN. After the data was collected we looked at Joost's application, network, and peer behavior.

Using the data collected from these test nodes, we performed various experiments to shed light on the questions we had about Joost. We began performing experiments to yield information on application behavior. Joost is based on XULRunner, a Mozilla runtime package, and uses the CoreAVC H.264 codec. By examining packets, we determined that Joost uses encryption for its control packets. We also found Joost's connection sequence, how it determines version through an HTTP GET request, and how reconnections are performed. Joost server infrastructure was also uncovered from extensive research of connection sequence IP addresses and various web sources. Experiments were performed to understand Joost's behavior when specific ports were blocked. Packet size data was also analyzed to determine control packet and video data packet ratios. Joost packets are 55% video traffic while the rest is control and acknowledgment data.

Analysis was then performed on the collected data to determine network behavior and how Joost usage affects a network. Our measurements concluded that inbound and outbound traffic remains consistent regardless of connection type. Network traffic was concluded to be reliant on Joost content servers as opposed to local network peers.

In terms of peer behavior, we found no correlation between the distance between two peers (geographical, RTT, or hop counts) and the amount of data transferred between them. We did noticed, however, that there may be some type of RTT-savvy selection algorithm at work, which gives priority to peers with RTT less than or equal to the RTT of a Joost content providing super node.

To conclude, Joost relies heavily on centralized content servers to provided licensed content while using the peer-to-peer overlay to service content at a faster rate. We argued that this approach does not scale well, and is sustainable today only because of the relatively low user population. Therefore, we predict a more distributed architecture in the future that takes more advantage of peer's proximity.

References

[1] S. Ali, A. Mathur, and H. Zhang. Measurement of Commercial Peer-To-Peer Live Video Streaming. In *Workshop in Recent Advances in Peer-to-Peer Streaming*, August 2006.

[2] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In *Proceedings of SOSP*, 2003.

[3] J. Liang, R. Kumar, and K. W. Ross. The KaZaA Overlay: A Measurement Study. In *Computer Networks Journal (Elsevier)*, 2005.

[4] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proceedings of IPTPS*, February 2006.

[5] S. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol, *http://arxiv.org/abs/cs/0412017v1*, 2004.

[6] B. Cheng, X. Liu, Z. Zhang, and H. Jin. A Measurement Study of a Peer-to-Peer Video-on-Demand System. In *Proceedings of IPTPS*, February 2007.

[7] V. Paxson. Strategies for Sound Internet Measurement. In *Proceedings of ACM SIGCOMM Internet Measurement Conference '04*, Taormina, Italy, November 2004.

[8] Joost FAQ, *http://www.joost.com/support/faq/*.

[9] "Los Angeles now serving the website and Long Tail Storage", *http://www.joost.com/blog/2007/04/los-angeles-now-serving-the-website-and-long-tail-storage.html*.

[10] “Good Day from London!”, *http://www.joost.com/blog/2007/04/good-day-from-london!.html*.

[11] Mozilla XULRunner, *http://developer.mozilla.org/en/docs/XULRunner*.

[12] “Joost got official UDP and TCP ports 4166 assigned by the IANA” *http://www.joostteam.com/2007/04/23/joost-got-official-udp-and-tcp-ports-4166-assigned-by-the-iana/*, April 2007.