AC 2008-881: CLASSROOM EXPERIENCE OF PEER-TO-PEER NETWORK TECHNOLOGY AS NEXT GENERATION TELEVISION

Veeramuthu Rajaravivarma, SUNY-Farmingdale

V. Rajaravivarma is currently with the Electrical and Computer Engineering Technology at SUNY, Farmingdale State College. Previously, he was with Tennessee State University, Morehead State University, North Carolina A&T State University, and Central Connecticut State University. Dr. Rajaravivarma teaches electronics, communication, and computer networks courses to engineering technology students. His research interest areas are in the applications of computer networking and digital signal processing.

Classroom Experience of Peer-to-Peer Network Technology as Next Generation Television

Abstract

One of the more challenging aspects of undergraduate Electrical and Computer Engineering Technology program is to bring the state-of-the-art technology experience into classroom. For many students, the traditional lecture/exam format is not effective at instilling the key concepts such that students truly understand. In the Digital Communication course during 2007, a new technology application class project called Joost "Bring TV to the Web" was introduced and received positive student responses. This paper describes the details of the class project information that can be integrated into any Networking or Telecommunications courses. The first part of the paper will introduce the ideas and business models behind Joost. It will discuss what makes Joost different and its advantages and potential disadvantages over its rival technologies. Then it will address the new P2P network technologies discussed in the class used by Joost and other important technologies implemented like H.264 for encoding and decoding and X.509 for encryption. The second part of this paper will focus on classroom experiment of the peer-to-peer network technology as a TV. As a result, students (1) will learn how to install and setup Joost application as a TV; (2) will observe bandwidth requirements, type of protocols used and the quality of the signal; (3) will learn how to use two open source software applications, Ethereal and Netpeeker, to measure the bandwidth of the signals; (4) will learn a new testing procedure. Results of Ethereal based experiment to document the protocol traffic and Netpeeker based experiment to get the maximum upload and maximum download rates are also tabulated in the paper. Finally student feedback and conclusion are provided.

Motivation

It wasn't long ago that TV networks started streaming selected series on their own sites. TV episodes are spreading not only to well-known portals such as AOL and Yahoo, but also to new destinations that mix in video searching, TV listings, and social networking. USA Today January 22, 2008 article on "More than just TV on the Web" reported that there are many video-rich web sites launched everyday, like Fancast.com, Hulu.com, Joost.com, Veoh.com, MeeVee.com, and MySpace.com/primetime. Some of the most notable competitors include YouTube and Apple's iTunes which uses solutions and infrastructure from Akamai. Also, many broadcasters such as Fox have streaming content right from their websites. Movies will soon be streamed from websites such as Netflix. Each uses a slightly different approach; each has a different business model. Most do use a similar architecture to stream video by means of unicasting or multicasting. This technique is generally efficient, but can tax content servers as more and more people request content. From a technology point of view, Joost takes a different approach to providing the video content using its P2P technology.

The main concern that led us to develop the new class project was constant difficulties that students were experiencing during the Networking and Communications course of loading and installing different software tools to test the quality of the signal. Even some of the step-by-step procedures proved to be cumbersome and time consuming to implement for many students. Furthermore, the examples included in most existing software toolkits often provide no insight as to what is happening and how each node is contributing to the overall execution of an application, where the bottlenecks are, and how system constraints and modifications can impact the performance. Considering all these shortcomings, although there are no data comparing the performance of Joost and other existing software toolkits, we believe, for the purpose of teaching,

this class project serves as an excellent tool. Moreover, flexibility of Joost application software allows students to use it outside the classroom environment and develop their own what-if scenarios. In this paper we describe a number of video quality test cases for laboratory experiments, which appeared to be very attractive to our students.

Business Model

Janus Friis and Niklas Zennstrom built their first peer-to-peer (P2P) application, Kazaa. The application, designed to allow the sharing of most any file type over an IP network, followed Napster as a way for people to illegally share digital music files. Rather than back down from the onslaught of litigation from the music industry, they fought, though they were eventually put out of business. It was a lesson learned for the two, but they certainly didn't seem to have turned gun-shy because they went on to tackle a second Goliath in the telecom industry. Skype, built on their improving P2P architecture, enabled the transfer of voice data across an IP network and interfaced with the publicly switched telephone network (PSTN). The result was a voice connection of acceptable quality with a significantly reduced cost or even free if both end points were Skype peer nodes. Skype was a major success as exemplified by the payday the two received when eBay purchased Skype for nearly $3 billion. With this established track record of tackling the Goliath industries and, with Skype at least, succeeding, the team of Friis and Zennstrom weren't happy to rest on their laurels. While still serving on the staff of Skype, the two privately financed the beginnings of a secret project that they dubbed the 'Venice Project'. Moved from the shadows as it was made ready for beta testing, the Venice Project, now named Joost, is a P2P video streaming application aimed at delivering television and movie content over an IP network.

Concept

With Joost, anyone will be able to search for and watch all their favorite shows, as well as some content that maybe wouldn't be available, over the Internet. Joost uses same hybrid P2P architecture used by Skype (licensed from eBay). The original concept was a part of the Kazaa specification [1] but available bandwidth technologies at that time were not up to the requirements to make the idea successful. The idea was shelved in lieu of the emerging demand for MP3 format file sharing optimizations. Joost offers additional features [2] to the viewer beyond traditional television including:

- On demand programming
- Program control with fast forward, reverse and pause.
- Current program information
- Search
- IM and chat rooms

Advantages

Of course, technology is all well and good, but ultimately it is the user experience and the market advantages that both viewers and content providers will react to. Joost has a different business model from its main competitors and this is what will separate it from the others as a choice for consumers and content providers. Joost has chosen to parallel the business model of traditional television by including advertising time for delivered content. For viewers, much of the content will be free just like traditional television. The viewers get additional features as described above, plus, certainly, a greater backlog of favorite shows and more control over what they watch than what is offered by traditional television or even many of the other competitors.

For content providers, the advantage comes in the form of more specific information about who is watching, when they are watching and where they are located. This will allow them to develop better trending models for class of content. Also, with this information, in-stream advertising can be targeted much more effectively than common television. This allows Joost to limit the amount of advertising time per hour of content. They estimate only 1 minute of advertising per hour of content. The return based on this targeted approach is anticipated to be greater. That's a plus for the advertisers and content providers. Viewers get to see fewer commercials and the ones they do see are more relevant to them. That's a plus for viewers.

Loading of content is controlled and is limited to production companies and other professional content makers. This makes the general quality of content similar to that of television, which is one of the biggest advantages of television. Also, it allows production companies to define geographic limits of release of their content. Other features for providers include more pay options such as pay-per-view and subscription schemes.

Content

In this type of application, simply building the technology is not enough. Content becomes the make-or-break component. Content providers will be wary about committing their content until Joost has become a proven medium with a considerable viewer base. Viewers won't come until there is enough content to make it worthwhile. Current selection is somewhat small, but will obviously grow as the concept proves itself. One thing that will help is the recent Viacom deal which will add content from Viacom's subsidiaries including MTV, VH1, BET, Comedy Central, Nickelodeon and Paramount Pictures. With a commitment from this large provider, it should increase Joost's momentum and continue to grow in terms of available content. As with other streaming video solutions, content comes from stored files provided to the network of users for download. As mentioned before, Joost does not have, nor were any plans mentioned for, a facility for live video or local programming. Obviously, this is an advantage of broadcast television.

Architecture and Technology Overview

Open Source

Much of the Joost application architecture has been developed using open source technologies and standards [3]. There are significant advantages to using this approach. Many problems in the computing domain have already been solved and solutions are available in the public domain as open source code. Using these open sources for the more general problems, it frees the development team up to put effort into solving the unique problems that the application needs to solve. Both time and effort are saved. Also, as issues are found and fixed in the open source code, the benefit of those fixes becomes immediately available. Of course, it becomes the development team's responsibility as a user of this code to contribute their own fixes to the open source code base if issues are found. It's not a requirement, but as part of the community and as a good neighbor, it's important that everyone contribute back to the community.

Another advantage to using open source technologies and standards is that it opens the application up to further extensions from others. Using popular technologies creates an instant pool of developers who know how to develop tools that will work with the application. While Joost will still have a significant amount of proprietary code, much of its general function will be open and will follow accepted standards. This is an important aspect of what Joost wants to become. Joost envisions, once it becomes a stable platform, that outside developers will code up their own widgets to include added value to the application. There are many examples of how this approach can gain significant momentum for an application. Most computer aided design

(CAD) programs such as AutoCAD or Microstation provide API's which allow users to write their own solutions if the application doesn't do exactly what they need it to do.
The client application uses open source code from:

- Cairo – 2D graphics library
- Mozilla - XML User Interface Language (XUL) Runtime Engine for SIP
- Redland – structured Web data using Resource Description Framework (RDF) specification
- SQLite – lightweight database engine

The server infrastructure uses open source code from:

- Apache HTTPD – HTTP Web server
- Jetty – Java-based dynamic content server
- OpenSSL – open source SSL implementation
- PostgreSQL – open source database
- Ubuntu Linux – Linux-based operating system

Many others, like Subversion for source code version control, are used to support the overall development process.

Hybrid P2P

Joost is built on a hybrid P2P network technology known as Global Index. Global Index is a proprietary technology developed by Joltid for the KaZaA file sharing application. This technology, also used by Skype [4], is licensed from eBay. Most communication software uses a central directory to track IP locations of and establish connections between nodes. This becomes prohibitively expensive when it has to be scaled to many millions of users. P2P is a good solution for decentralizing the network management. Typical P2P networks, however, are fragmented in nature, thus making it impossible to find every node with any particular search. Global Index uses a system of multi-tiered 'supernodes' that communicate with each other to maintain connectivity for the full network. Supernodes are the main contact point for the set of nodes that it services. Requests for a piece of content will start at the supernode and be passed around the network of supernodes and on to the leaf nodes until the requested content is found. Any node in the network is eligible to be elevated to a supernode and the status can change over time and as nodes go offline or come online (see Figure 1 and Figure 2). Figure 2 shows that when a supernode goes offline, it is possible for another to take its place with little disruption. The Global Index network works across NAT (Network address translation) and firewalls by using other nodes on the network as proxies to increase the chance and quality of connections. Only proxies with available resources are used for this purpose ("Skype Explained", n.d.[4]).

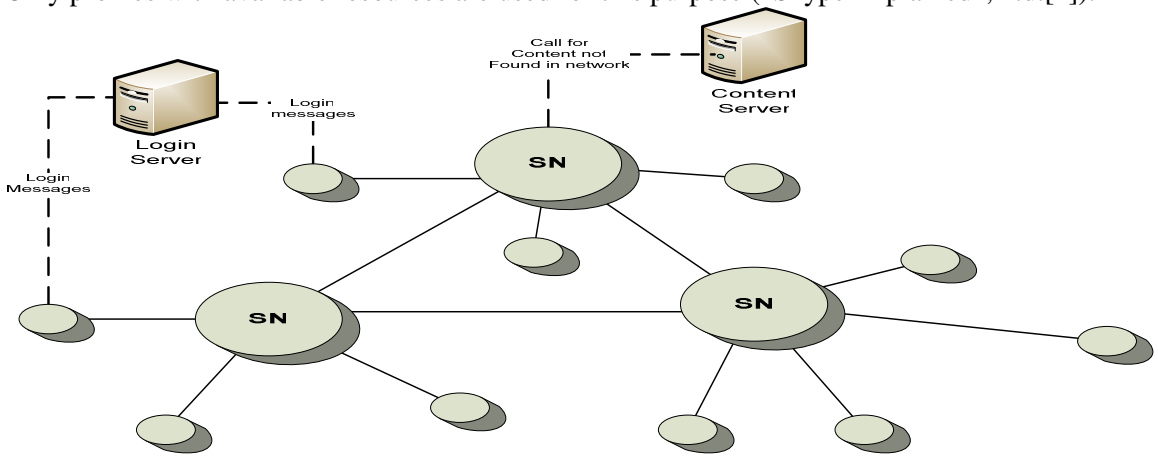

Figure 1 Joost P2P Network

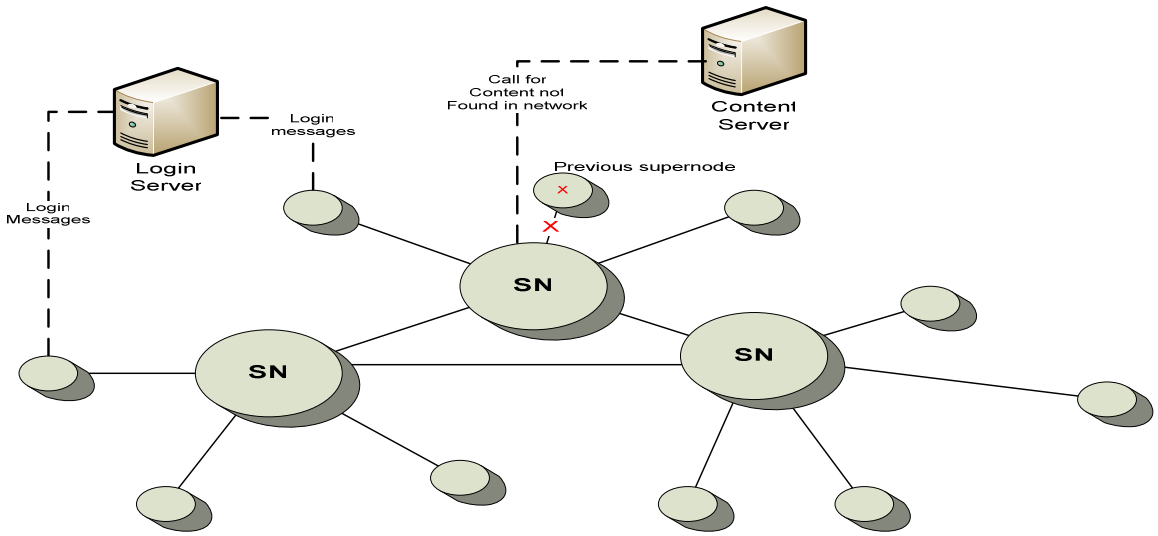

Figure 2 Joost P2P Network - Updated Supernode

Other key components of the network infrastructure used by Joost include support facilities for security and for initial content seeding. A login server controls access to the network. Upon startup, the user ID and password are authenticated and the node advertises itself to its local P2P network nodes. Also during the login process, the type of NAT and firewall is determined and nodes with public IP addresses are determined. These are used in case the current node's super node cannot be contacted at any point [5]. Content servers, dubbed 'long tail storage', seed the network with content and ad content. Content snippets, including ad content, are stored locally to be provided to others in the network upon request. If a piece of content being requested is not found in the network-at-large, then the content server is the last stop as it is guaranteed to be available on this server. A backend ad engine pinpoints viewers by location, time of day, viewing habits and opt-in profile ad information and provides the targeted advertisements.

Video Streaming Solution

Joost achieves streaming in a P2P environment by caching content on users' machines in ~10 second snippets. This content is initially seeded from Joost's content server. An Ad engine injects the targeted ads along with the seeded content. Snippets of both content and advertisements are stored locally on users' machines waiting for the next user to request that content. When a segment of content is required at the client it requests it from the network at large. If that segment is not available, the content server must be contacted to supply that segment. If it does find the requested snippet, a UDP message is used to send the snippet from its cached location in the network to the requesting machine. Depending on it's availability in the network, that snippet may or may not be cached permanently on that machine to increase the chances of the next requesting machine being able to successfully find that snippet.

Reiss [1] gave the following description (with Figure 3 below) of Joost's process:
"After Joost makes a show available, the first users to request it (A) query the network at large (B) to see whether peers can provide the program. If they can't, the request goes to a content server (C), which streams the show, interspersed with individually targeted ads, directly to each viewer's screen (D). While the users watch, short segments of the show and the ad are saved to their local hard disks (E). In this way, the entire show and a variety of ads are seeded throughout the network. So when another user (F) requests the same title, that show, along with a targeted ad, comes not from the server but from the network, one segment at a time. Once again, fragments of the show and ad are stored to the new user's hard disk (G), ready to be streamed to others. - Ted Greenwald" (Reiss, 2007, p. 96 [1])

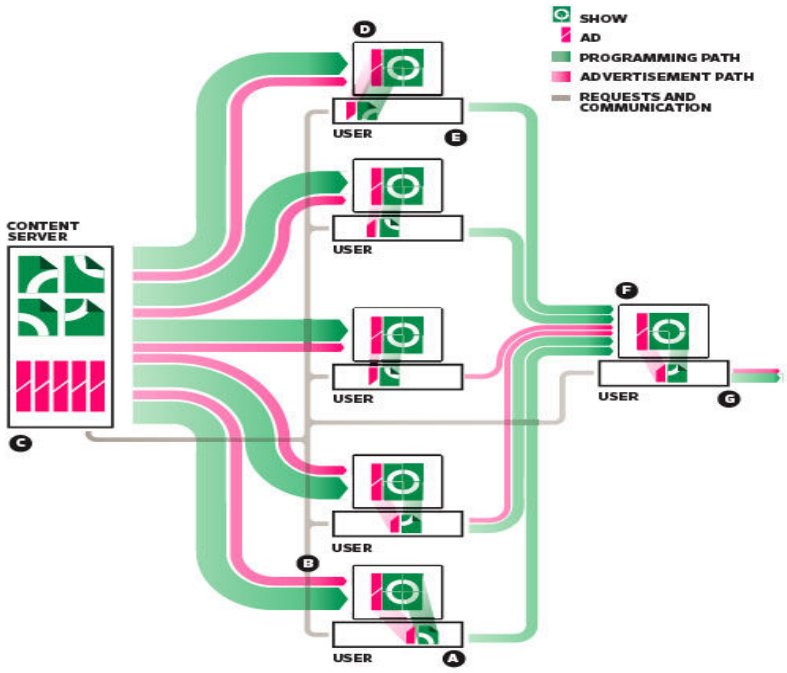

Figure 3 Streaming with Ad Content (Reiss, 2007 [1])

Ad content is also broken up into segments and cached on local hosts throughout the network. It's unclear if Ad content is managed as completely separate video content to be requested from the network at large or it's strictly tied to a piece of television content. In either case, because Ad content is cached throughout the network, it gives Ad content a much longer lifespan than might be seen on traditional broadcast television. The network will constantly be updated with new targeted Ads, but will also remain seeded with previous Ads. Updating, or removing, a particular Ad was not an issue that seemed to be addressed in any literature published yet.

Protocols

Joost makes use of both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) depending on the function and situation that it is trying to accommodate. UDP is used for the actual video streaming data because of its improved speed over TCP. UDP requires so little overhead that it is used when speed of packet delivery matters more than the assurance of packet delivery. The Joost website describes its use of the following protocols and ports used [6]-[8] ("Joost FAQ: Problems", n.d.):

- UDP for video streaming
- TCP for P2P management
- TCP access, outbound, to ports 80 (HTTP), 443 (HTTPS) and 5223.
- UDP access, outbound, to port 33333.
- If you have a public IP address and UDP inbound on port 33333 is unfiltered, the client may use UDP on port 33333 inbound.
- If you want to chat from within Joost, you'll also need to enable inbound TCP access to port 5223.
- Check also that your network has DNS servers configured - Joost client makes regular DNS requests like any other application.

H.264 Codec

Joost's content is processed to be rendered at DVD-quality at 400 Kbps leaving room for upload bandwidth with the typical 500 Kbps home connection. This is done with the H.264 standard, also known as ISO/IEC MPEG-4 Part 10. H.264 was finalized in May 2003 and is a jointly maintained standard from ITU-T Video Coding Experts Group (ITU-T VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) [9] ("H.264/MPEG-4 AVC", n.d.). The Wikipedia H.264 topic goes on to say:

> *"The intent of the H.264/AVC project was to create a standard capable of providing good video quality at substantially lower bit rates (e.g., half or less) than previous standards (e.g., relative to MPEG-2, H.263, or MPEG-4 Part 2), without increasing the complexity of design so much that it was impractical (excessively expensive) to implement. An additional goal was to provide enough flexibility to allow the standard to be applied to a wide variety of applications (e.g., for both low and high bit rates, and for low and high resolution video) and to make the design work effectively on a wide variety of networks and systems (e.g., for broadcast, DVD storage, RTP/IP packet networks, and ITU-T multimedia telephony systems)."* [9] ("H.264/MPEG-4 AVC", n.d.)

One of the biggest advantages of H.264 over previous standards is its compression performance. Compared with standard DVD (MPEG-2), H.264 can deliver better image quality at the same bitrate or lower bitrate for the same quality image. A single-layer DVD can hold a 2 hour movie in MPEG-2 format while the same disk could hold 4 hours in H.264 format or the same 2 hour movie in much better quality [10] (Richardson, 2007). For a further example of the compression rates that H.264 can achieve, a tutorial on Alex Mejia's Blog [11] ("Alex Mejia's Blog", March 7, 2007) gives an example of a 35 second AVI file that is 111 MB in size compressed down into a 2 MB MP4 H.264 format encoded at 500 kbps. The tradeoff for this improvement is that the encoding process is very intensive and takes a significant amount of time. However, the result is a very highly compressed, very high quality encoded bitstream that can be streamed over an IP network. That encoded bitstream is then decoded at the client machine to display the content. Joost uses a product called CoreAVC [12], created by CoreCodec, to do H.264 coding and decoding [8] ("Joost FAQ: Technology", n.d.). The CoreAVC High Definition H.264 product is based on the MPEG-4 Part 10 standard and is one of codecs used by both next-generation DVD formats: Blu-Ray and HD-DVD. CoreAVC implements a software video decoder that they say is faster than other solutions that rely on hardware to increase playback performance of H.264 video. CoreAVC, which handles the audio\video compression, includes the Haali Media splitter [13] to separate audio, video and subtitles into separate streams for storage and delivery of a complete movie or CD in a single file called a Matroska container. Matroska is an open standard format based on Extensible Binary Meta Language (EBML) which is a binary derivative of XML. Matroska aims to be an open source alternative to competing containers such as AVI, MPG, MOV and MP4. It has features such as [14] ("Matroska Official Homepage", n.d.):

- Fast seeking in the file
- High error recovery
- Chapter entries
- Selectable subtitle streams
- Selectable audio streams
- Modularly Extendable
- Streamable over internet (HTTP and RTP audio & video streams)
- Menus (like DVDs have)

Joost is using a technology that's been proven to work in both next-generation DVD formats. Following its intent, this technology is also serving as an excellent format for transmission of high quality video over the Internet.

Encryption

Joost, and indeed any application distributing proprietary content over the Internet, faces security issues from two fronts. On one hand, because of its P2P nature, users have concerns of receiving a tremendous amount of IP traffic from many different source machines. It could open up a gateway for malignant code to infect a user's system. On the other hand, content providers become very nervous when their wares are distributed over the internet. It could be pirated, copied and distributed in a way that would not allow them to control their revenue stream. To answer these security concerns, Joost decided to encrypt all their traffic using the X.509 standard.

X.509 is a public key infrastructure (PKI) standard developed by ITU-T. X.509 defines formats for public key certificates and a certification path validation algorithm. Version 3 of X.509 includes support for topologies such bridges and meshes meaning it can be used in a P2P architecture [15] ("X.509 Wikipedia", n.d.). This makes it an acceptable solution for Joost's requirement of securing their transmitted content. Implementation of an encryption scheme is one of the main things that will make Joost a viable solution for content providers. Joost encrypts content streams to be piracy- protected. This is attractive to content providers looking to take advantage of digital media, but who are concerned about illegal sharing. Just as important is the user base's comfort with the security of the application. X.509 answers the concerns of both parties nicely.

Classroom Observations

Quality

Quality of streaming video is judged based on the smooth delivery, on the speed of delivery of the content and on the quality of the picture and sound delivered. Many variables affect the quality of a multimedia system such as Joost. Bandwidth availability to download a piece of content and content availability are both key components. Beyond the logistics of delivering the files over an IP network, the encoding/decoding process also affects the apparent quality of the program.

With traditional client-server architectures, being able to scale up to demand is an ongoing issue: the more people making requests, the more the drain on the system and the more the risk of system failure increases. Denial of service attacks are built on this principle. There are many different solutions to solve this problem in a client-server architecture such as load balancing of redundant servers to support load distribution and failover. Joost's P2P architecture, at least theoretically, does not face this same shortcoming. Certainly, P2P has its own set of difficulties and complications, but, in the terms of increasing loads, it excels compared to traditional systems. P2P quality typically increases as more users request a piece of content. This is because more and more copies of the files, or snippets in Joost's case, are distributed through the system as more people watch that piece of content. As availability increases, a requestor has more choices of where to acquire a particular snippet and so the fastest available connection can be chosen for download. So, content availability becomes less of an issue as more people want to watch it; quite the opposite of client-server systems.

Another aspect of quality is a smooth delivery for the streamed video. This is heavily dependent on the bandwidth. It was observed that using Joost on the test machine on a DSL connection over wireless operating at 0.6 Mbps, programs still tend to stall during play. This is especially apparent when an advertisement is being retrieved from the network. It seems to stall for 10's of seconds as it retrieves a 10 second advertisement. This will obviously not satisfy viewers. It was observed and other articles and blog entries anecdotally mention that picture quality is not quite as good as it needs to be to compete with broadcast television. As mentioned earlier, the Joost website says it encodes content to be rendered DVD quality to play a show on full screen mode at

1600 x 1200 resolution really displays a significant amount of pixilation. Admittedly, the video card that this observation was made on is not anywhere near top quality. To truly get the full benefit of the encoding process, a high quality video card with onboard video memory is required. During install, a screen is displayed noting for the user any deficiencies in their system based on Joost's minimum requirements.

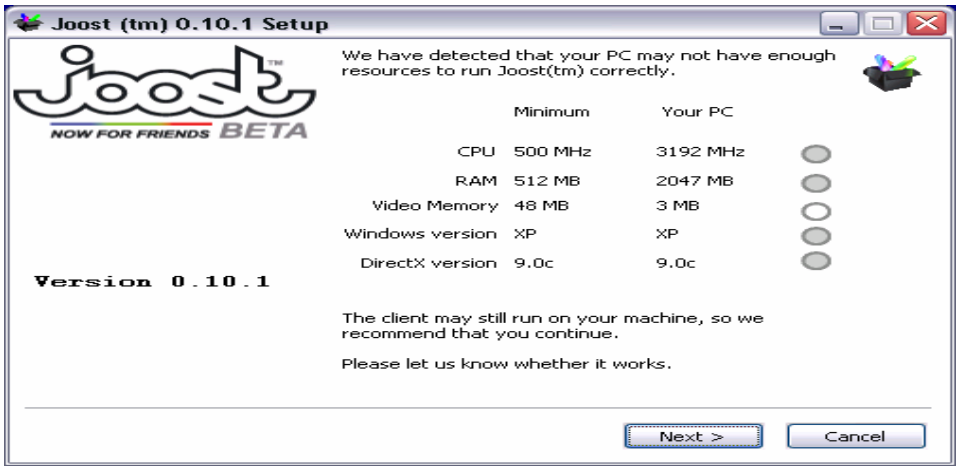

Figure 4 Joost Install - Minimum Requirements

Bandwidth Requirements

The Joost website estimates that in 1 hour of viewing 320 Mb of data is downloaded and 105 Mb uploaded [7] ("Joost – FAQ: Using", n.d.). That is 0.0889 Mbps download and 0.0292 Mbps upload for a total of 0.1181 Mbps of required bandwidth. Online speed tests [16] (http://www.speedtest.net/) put the test machine's DSL connection at somewhere between 0.5 Mbps and 0.7 Mbps for download and 0.1 Mbps for upload. This DSL connection seems to reasonably exceed the estimated required bandwidth.

Measurement Analysis

Two applications were utilized to measure the bandwidth used. Ethereal [17] (www.ethereal.com) was used to document the protocol traffic in detail. Netpeeker [18] (www.net-peeker.com) was used to get the maximum upload and maximum download rates. Measurements were taken during the following conditions:

- Program play
- Play paused
- Program play after pause – Program is paused, but the application is left open.
- Application minimized – The application is still running in the system tray.
- Program play for second time (same session) – The same program is played again during same application session.
- Program play for second time (next session) – The same program is played again after the application is completely shut down and restarted.

The first few attempts to use Joost resulted in a series of Program Unavailable and Network Disconnected messages. This seems to have indicated some issue either with Joost's network or with the network at the test machine. Testing later continued unimpeded by these issues.

Program play shows a significant amount of UDP traffic from many different IP addresses over the 10 minutes of play time measured. As the program was paused, download continued and upload to the Joost network actually elevated. It seems reasonable to assume that the user will continue to watch the show if they pause it, so continuing to download the content does make

sense. After the program was restarted after the pause, download of snippets continued and upload tailed off. The application was then minimized to the system tray. It was observed that download continued at a steady rate. When the application is restored from the system tray, the program is restarted where it was stopped. While continuing to download during program-paused state is a reasonable decision, continuing to download during system tray operation seems to be a poor use of bandwidth. There's no guarantee that the user will continue watching the program any time soon, nor that they will want to continue watching the same program when they do reopen the application.

When the same program was restarted in the same application session, it was observed that the quality of the program was greatly improved. Fewer pauses were noticed during the play. Download was greatly reduced, also. This seems to point to the fact that program segments already played are cached for some time on the host machine. The same program played in a different application session did display less download bandwidth, but it was not a huge decrease. As discussed earlier, some program snippets are cached locally. The application checks locally to see if a required snippet exists before requesting it from the P2P network.

After Joost is completely shut down, some traffic is still seen. This appeared to be requests for snippets from other nodes in the P2P network. Apparently the loss of the test machine as an active Joost node is not broadcast to the rest of the P2P network right away. Measurements detailed in Table 1 show that on the peak download rates corresponded to the expected rate of 0.0889 Mbps. However, the peak upload rate did not even approach the expected rate of 0.0292 Mbps. The average bandwidth used was much less than expected. The program play was fairly choppy overall. There was no indication of why Joost wouldn't use the full 0.1181 Mbps if that bandwidth was available. There is a possibility that one of the tools used for measurement, Netpeeker, actually inadvertently limited the speed. The trial version of this tool expired before the measurements were made. While the facilities that were used to get the maximum upload and download still worked after the trial period ended, one of the features that was disabled was the ability to configure the speed limiter. It's possible that, even in expired-trial mode, this tool was limiting the speed. However, at the time of writing, there was no proof available that this was the case. UDP was the protocol most used during program play. Figure 5 shows the number of UDP and TCP packets received over time. Program snippets during active watching of new content came in from 162 different IP addresses over the course of 10 minutes (see Table 1). TCP was really only used for communication with Joost's backend servers.

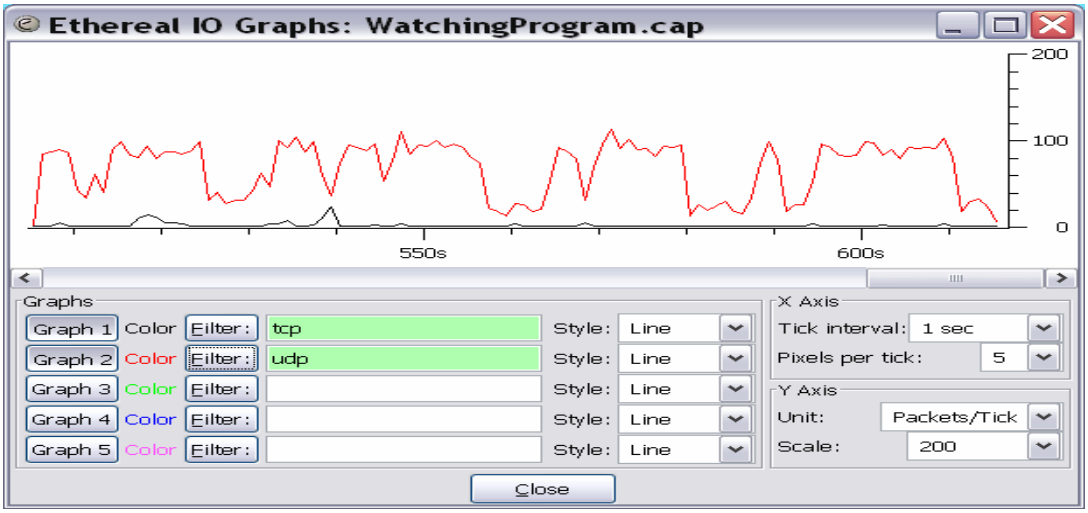

Figure 5 UDP and TCP Packet Traffic During Program Play

Situation	Peak Upload (Kpbs)	Peak Download (Kbps)	Time Observed (Min:Sec)	Total Traffic (Kb)	Avg. Bandwidth Used (Kbps)	Number of Conversations (TCP/UDP)	Number of UDP Endpoints (Unique/Total)	Number of TCP Endpoints (Unique/Total)
Baseline	0.107	0.103	5:12	1.914	0.00612			
Unavailable program message	23	89	3:38	10,583.1	48.51	42/383	259/387	5/46
Actively watching new content	4.9	88.5	10:16	40,982.7	66.51	7/216	162/224	3/9
Content paused	8.14	88.00	8:14	31,371.4	63.5	4/256	197/262	3/7
Actively watching after pause	0.56	87.3	6:00	23,167.2	64.21	56/131	122/136	4/60
Application minimized to system tray	2.65	86.3	12:05	38,141.5	52.54	2/217	166/225	3/4
Watch content 2nd time (same session)	1.78	2.70	10:00	721	1.2	4/107	102/114	3/7
Watch content 2nd time (next session)	8.61	63.21	10:00	2,286.3	3.8	20/281	167/288	3/23
After exiting Joost	0.107	0.107	7:57	26.5	0.055	1/29	24/34	2/2

Table 1. Bandwidth Measurements

Student Feedback

Various methods were used to formally assess the effectiveness of this class project including the instructor's assessment of laboratory work and classroom presentation. Based on the student's feedback, the overall response from students regarding whether the class project met their expectations was very positive and the project integrated into the course was positive. Summarized student comments are:

- This course presents interesting topics and help them to learn new technologies
- They have a better understanding of Networking and Telecommunication software testing tools
- They feel confident to integrate Networking and Telecommunication Technology in their senior design projects.

In addition to the semester end course evaluation, students were requested to submit their course feedback via online by the end of the semester. Results of students' feedbacks for 10 questions are shown below:

Question #1: Why did you sign up for this course?
Responses varied from a required course in curricula to students who wished to learn and get hands-on experience with network and telecommunication tools.
Question #2: What are the reasons for your attendance or the lack of your attendance?
Students responded that class was interesting and weekly exams provided motivation. Sickness, conflicts and attitude were blamed for lack of attendance.
Question #3: How did you cope with the course assignments?
Some students noted that with study and balancing other requirements that they could cope, but a few commented on difficulties.
Question #4: How hard or easy were the quizzes?
Students' response depended entirely on the individual chapters. Some chapters were simpler than others; some were long and complex. In general the quizzes seemed fair for most of the students.
Question #5: Were the sessions sufficiently prepared by the lecturers?
Everyone in the class felt that the lecturer prepared the students to the best of his ability.
Question #6: Were the topics presented in a confusing way? Why or why not?
In general, students felt certain topics were confusing, but they were often presented in a way that made them seem simpler by comparing them to real life situations.
Question #7: Did you read the assignments? Why or why not?
Most students read their assignments
Question #8: What do you think about the course content?
Students felt the course material was interesting
Question #9: What topics are missing?
Most students felt that the coverage was complete.
Question #10: What would you like to know more in addition or instead of which course topic?
Students would like to learn more about real life networking applications such as security, physical networks, and VOIP. Also received are comments on the improvement of the course, mainly on how to incorporate more practical application projects into the laboratory.

Conclusion

In this paper we presented the new P2P networking technology and the practical demonstration of the P2P technology of TV on the Web in a junior level Digital Communication course. We introduced a simple and easy-to-use Windows-based graphical Joost application software toolkit.

Joost allowed students to easily install, setup, and create a Web TV anywhere in the world using wired or wireless internet connectivity. This flexibility of Joost allowed students to use it outside the classroom environment and develop their own what-if scenarios. In this paper we also describe a number of test cases for laboratory experiments. These test cases can easily be extended to examine more complicated and advanced scenarios of other websites mentioned in the paper to compare the streaming video quality.

Bibiliography

[1] Reiss S., (2007). "Here comes trouble", *Wired*, Feb 2007, pp 94-101

[2] What's Joost? (n.d.). Retrieved March 1, 2007, from http://www.joost.com/whatsjoost.html

[3] Joost – Open Source. (n.d.), Retrieved March 15, 2007 from http://opensource.joost.com/

[4] Skype Explained. (n.d.). Retrieved March 1, 2007 from http://www.skype.com/products/explained.html

[5] Baset, Salman A. and Schulzrinne, Henning (2004). "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol", Retrieved on March 15, 2007 from http://www1.cs.columbia.edu/~library/TR-repository/reports/reports-2004/cucs-039-04.pdf

[6] Joost – FAQ: Problems running Joost. (n.d.). Retrieved March 15, 2007 from http://www.joost.com/support/faq/Problems-running-Joost.html

[7] Joost – FAQ: Using the client. (n.d.). Retrieved March 1, 2007, from http://www.joost.com/support/faq/Using-the-client.html

[8] Joost – FAQ: Technology. (n.d.). Retrieved March 1, 2007, from http://www.joost.com/support/faq/Technology.html

[9] H.264/MPEG-4 AVC. (n.d.), Retrieved April 28, 2007 from http://en.wikipedia.org/wiki/H.264

[10] Richardson, I. (2007). "An Overview of H.264 Advanced Video Coding", Retrieved April 28, 2007 from http://www.vcodex.com/h264.html

[11] Alex Mejia's Blog: How to encode h.264 files for FREE with MeGUI. (March 7, 2007). Retrieved April 28, 2007 from http://alex.nigma.info/2007/h264-tutorial/

[12] CoreAVC.com. (n.d.). Retrieved April 28, 2007, from http://www.coreavc.com/

[13] Haali Media Splitter. (n.d.). Retrieved April 28, 2007, from http://haali.cs.msu.ru/mkv/

[14] Matroska Official Homepage. (n.d.). Retrieved April 28, 2007, from http://www.matroska.org/

[15] X.509 - Wikipedia. (n.d.). Retrieved April 28, 2007 from http://en.wikipedia.org/wiki/X.509

[16] Broadband speed test: http://www.speedtest.net/

[17] Network protocol analyzer: www.ethereal.com

[18] Network monitor, firewall and speed limiter: www.net-peeker.com