

Measurement of Voice Over IP Traffic

J.P. Curtis, J.G. Cleary, A.J. McGregor & M.W. Pearson

Abstract— A procedure for recognising Voice over IP (VoIP) traffic is described. It is shown how to apply this to traces of Internet traffic containing only header information. The process is heuristic relying on recognising both the TCP setup phase of H.323 sessions and the subsequent UDP traffic.

Measurements of VoIP traffic at a number of concentration points on the Internet are presented. These measurements are then used as input to simulations of mixed TCP and UDP traffic. These simulations attempt to answer the questions of how much impact unregulated VoIP traffic will have on TCP traffic.

Keywords— Voice Over IP, measurement telecommunications traffic.

I. INTRODUCTION

VOICE Over IP (VoIP) is a fast growing technology within the fast growing Internet. It allows voice conversations to be transmitted over the standard IP network. At its best, it provides quality better than the existing telephone network and at much lower costs. As a result of this VoIP traffic is growing rapidly, possibly faster than general Internet traffic.

VoIP is a real-time service, unlike the classical Internet applications, such as HTTP, FTP and SMTP. Most of these applications use TCP with its relatively well understood back-off and rate control mechanisms.

VoIP is usually implemented using UDP which can provide better real-time responsiveness and lower overheads. UDP based applications do not have to implement congestion control. Because of this, any significant increase in the proportion of UDP traffic gives cause for concern. The performance of today's TCP based applications could be seriously affected by such an increase.

To investigate these effects we have created a simulation of the interaction between TCP and VoIP traffic. This paper emphasizes the process of obtaining detailed traffic traces and their use to generate traffic within the simulation.

Unlike other traffic types VoIP cannot be simply identified by IP fields or by port usage. Also, because measurements have been taken on real operating networks, security concerns have meant that only header information is available not packet contents. Thus, identifying VoIP traffic is a non-trivial task. We describe the techniques we have developed to identify H.323 traffic which is the primary standard used for VoIP traffic. These look for signalling traffic over known TCP ports (to set up a conversation) followed by heavy usage of a pair of UDP ports.

J.P. Curtis, J.G. Cleary, and M.W. Pearson, Department of Computer Science, University of Waikato, Hamilton, New Zealand {jpc2,jcleary,mpearson}@cs.waikato.ac.nz. A.J. McGregor, National Laboratory for Applied Network Research (NLANR), San Diego Super Computer Center, 10100 Hopkins Drive, San Diego, CA 92186-0505, USA, tonym@nlanr.net

Results from checking passive measurement traces taken in New Zealand and the United States are presented. These show that the traffic densities and time variation of VoIP traffic is very different from dominant TCP traffic types such as HTTP. The paper concludes with preliminary results from simulations using models derived from our measurements.

II. TCP AND UDP

Today the Internet is experiencing one of the fastest growth rates of any technology seen before. Its audience has quickly spread from academics and computer scientists to the general public, and is now seen by many as an indispensable tool. For many users, however, both connection and machine speeds have limited the potential uses of the Internet. This is now starting to change with recent increases in machine performance and decreasing costs of broadband connections to the home. With these improvements it is becoming possible to provide a method of communication very similar to the standard telephone network using an Internet connection. The increase in bandwidth, when combined with the advances of audio compression and machine speed has allowed these services to be capable of high quality, often better than the current telephone network. Many users are also likely to accept a drop in quality for the benefit of free long distance calls that using the Internet provides. This being the case, an even wider range of people are able and likely to use these systems. It is not surprising therefore that Voice Over IP (VoIP), is an increasing use of the Internet today.

VoIP has strict real-time requirements on both jitter and delay if the transmitted speech is to be intelligible. Because the rate and congestion control algorithms of TCP tend to introduce both jitter and delay (but in return provide reliability) VoIP is usually implemented using UDP which eschews reliability to gain better real-time performance.

TCP, besides being responsible for error checking and correcting, is also responsible for controlling the speed at which data is sent. TCP is capable of detecting congestion in the network and will back off transmission speed when congestion occurs. These features protect the network from congestion collapse. TCP is a reliable service and delays are introduced because, whenever a bit error or packet loss occurs the broken packet is retransmitted. This can be a large source of jitter.

TCP uses a combination of four algorithms to provide congestion control; slow start, congestion avoidance, fast retransmit and fast recovery [7]. These algorithms all use packet loss as an indication of congestion, and all alter the number of packets sent before waiting for acknowledgments. These alterations affect the bandwidth available to the TCP connection and also change delays, providing

another source of jitter.

These combined effects make TCP unsuitable for real-time transmission. Voice communication, however, has the advantage of not requiring a completely reliable transport level. The loss of a packet or a single bit error will often only introduce a click or a minor break in the output.

For these reasons most VoIP applications use UDP for the voice data transmission. UDP is a thin layer on top of IP that provides a way to distinguish among multiple programs running on a single machine. UDP also inherits all of the properties of IP that TCP attempts to hide. UDP is therefore also a packet based, connectionless, best-effort service. It is up to the application to split data into packets, and provide any necessary error checking.

When UDP traffic is introduced into a congested link it will not back off, forcing any TCP traffic to back off further. The possibility of VoIP traffic causing a significant increase in UDP traffic provides motivation for this investigation into VoIP systems. The ultimate aim is to quantify the impact of UDP on TCP traffic.

III. SIMULATION

The simplest way of quantifying the impact of UDP is via simulation. Simulations can control parameters such as connection bandwidth, router queue lengths and traffic levels, and record statistics such as queue utilization, packet loss and connection throughput. Attempting to obtain these sorts of results from an actual network would be both extremely costly and difficult.

Simulation however can only produce results as reliable and as accurate as the models employed. For simulation results to have any meaning, the component models in the virtual network, (such as switches, routers and most importantly, traffic models) must be as close to the real-world as possible.

A simulator and traffic model exist for HTTP traffic which makes use of TCP for transmission [8]. These systems will form the base of a simulation for this investigation. The throughput of the HTTP sessions will be tested as varying UDP data rates are carried on the network.

Traffic models are based around network traces. The simulator takes data recorded from these network traces and replicates it during simulation to provide loads for the network. If there is insufficient data to create the required loads the simulator offsets and overlays multiple copies of the recorded data.

A. Data source

Two options are available to create the traces required by the simulator. The first is to accurately measure VoIP data in a lab environment. The advantage of this approach is complete control of the systems and protocols used. This means that there is no question what the traffic represents, what programs and compression techniques were used, or the speed and load of the machines and Internet connections involved.

The second option is to use network trace files that contain IP and transport layer headers for every packet at a

specific point in the network. Once collected the trace files can be checked for data that has the signatures of VoIP traffic. This filtered data then forms the basis for simulation.

The second option may seem at first more complex and less accurate than the first, but also has advantages. Firstly, there are many providers of VoIP applications and these can use different voice compression systems. Secondly, different users of VoIP applications may have varied Internet connection and machine speeds. Each user will therefore be able to handle different data rates and compression techniques. Lastly there is no way of knowing what a 'normal' conversation is like, a pair of research students may have completely different conversation patterns to a business meeting. All of these issues will alter the traffic created, both in speed and size. Trying to cover all of these possibilities in a lab environment would be infeasible.

Identifying actual data in use on real networks ensures that the data used in simulation is a close representation of the type of voice traffic seen at that point in the network. For these reasons it was decided to try the second alternative and detect VoIP traffic sessions using only IP header traces.

IV. TRAFFIC SOURCES

This project has drawn from two measurement projects to provide the network traffic traces. While both passive measurement systems, these two projects take different approaches, resulting in different strengths and weaknesses that have to be respected when drawing conclusions from the data. The remainder of this section outlines these projects in more detail.

A. The New Zealand Internet Exchange

The first of the two data sets comes from the *New Zealand Internet Exchange (NZIX)*. This set has been collected by the *Waikato Applied Network Dynamics (WAND)* [1] group, at the University of Waikato, New Zealand. This section aims to both give an overview of the measurement system, and details some of results of an initial exploration of the data set.

The NZIX is a 10/100 Mbit Ethernet switch located at the University of Waikato, New Zealand. It forms an inter-connection point for multiple New Zealand bandwidth providers and also provides the University's Internet connection. The WAND group have been permitted to collect trace files of the traffic flowing through this switch.

The NZIX measurement project was started on the first of December 1998 and ran until the end of March 1999. Each day, at 10am and 2pm NZST a 10 minute capture was taken. This has resulted in 211 trace files that, compressed, total approximately 6.4GB.

Shown in Fig. 1 is a plot of the daily traffic average, for the year covering the period of the project. This plot was taken from the SNMP data provided by the switch, related to the traffic on the SPAN port. This, and graphs of other ports of the switch can be accessed publicly from the WWW [2].

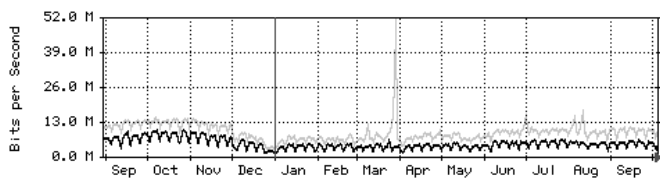


Figure 1. Years traffic plot of NZIX

The black line of the plot represents the daily average of the data rates. The gray line represents the maximum 5 minute data rate per day. It can be seen from this plot, and also from the trace sizes, that the data rates on NZIX have changed over the time of the project. They initially dropped, due to a breakup of the Kawahiko organisation, a large contributor to NZIX traffic. Data rates dropped to a low around Christmas, and then stayed consistent for the rest of the project.

The measurement makes use of the *Switch Port Analyser (SPAN)* port present on the NZIX switch. This port can be configured to mirror traffic on some or all of the other ports of the switch. In its current configuration the switch is copying all traffic flowing through the switch to the SPAN port. The process of copying data to this port can, however, interfere with the packet timestamping process and can also drop packets if the switch has more traffic flowing through it than the SPAN port can accept. As the data rates are significantly lower than the available bandwidth on the SPAN port it is not expected that packet loss is a problem. It is unknown what timestamp inaccuracies may be introduced by the switch.

The SPAN port is attached to a standard network card in a PC running Linux. Custom software, written at the University of Waikato, is used to capture and timestamp the packets. This software uses the `pcap` packet socket interface to the kernel network stack. Unfortunately such software monitoring systems have problems due to the fact that none of the components were designed or optimised for network measurement. There are many delays, and buffers in a software solution that reduce the accuracy of timestamps and increase the possibility of packet loss.

The first place this occurs is at the hardware level. A network card may not be reliable enough to capture every packet on a network. In normal use these dropped packets would be corrected by the TCP stack. In a monitoring situation, undetected packet loss could cause problems with analysis, or just reduce the accuracy of results. Another common problem is that in an attempt to increase performance, many network cards will buffer a stream of packets. The card will then only interrupt the host once and deliver a collection of packets, as opposed to once per packet. This will increase host performance, but will mean that the timestamps on successive packets may not represent the actual inter-packet time.

The timestamping process in a `libpcap` system occurs in the kernel. This timestamp can be delayed if the host system is under load, and the interrupt service for a packet is delayed. Also the buffers that store the packet before de-

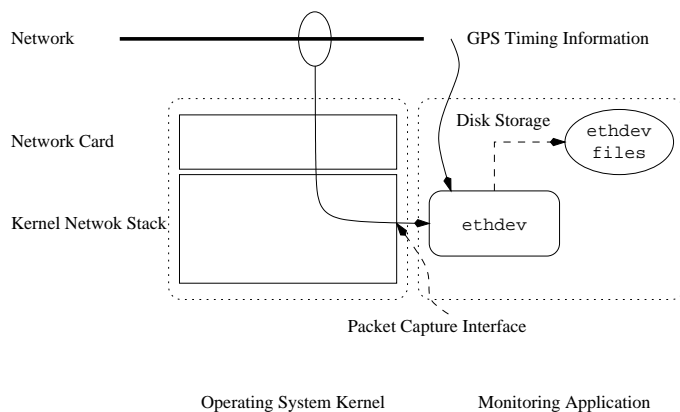


Figure 2. Ethdev measurement layout

livery to the `libpcap` application are unchecked. If an application program does not clear the buffer quickly enough, then packets can be silently dropped.

The measurement software, named `ethdev`, has been designed to be one part of a larger passive one way delay measurement system. Because of this, GPS support is included in the `ethdev` system. At one second intervals a zero length packet is inserted, along with the software timestamp at this point. This allows post processing of the files to correct for clock drift. While this greatly reduces the inaccuracy of the clock on the system, it does not reduce the delays before a packet is timestamped. A general layout of the `ethdev` measurement system is shown in Fig. 2.

Clearly care must be taken when drawing results from these traces. The accuracy of the derived results must take into account the margin of error present in the original measurement system, and must be kept in mind in the following sections.

The software records the first 54 bytes of each packet, made up of 14 bytes of Ethernet overhead, and 40 bytes of IP. Associated with each packet is a 32 bit 4 MHz timestamp, resulting in a timestamp wrap approximately every 17.9 minutes. Lastly the wire length of the original packet is also stored. If this value is set to zero, it indicates a GPS interrupt.

For security and privacy reasons, the source and destination IP addresses in the recorded trace files are encoded. The encoding is a replacement system, where every IP address is replaced by a false substitute. These substitutions are consistent over all of the traces. A database of the translation between substitute and original is held by the maintainers of the NZIX switch. Translations are released on request, when the owner of that IP address has provided consent for this to occur.

B. MOAT, NLANR

The Measurement and Operations Analysis Team (MOAT) are a division of the NLANR research team, in San Diego, USA [3]. The main aim of the MOAT team is to create a network analysis infrastructure, providing a platform to build a better understanding of the Internet. One

major part of this is a large scale passive monitoring system, that consists of multiple measurement points spread across the USA. These monitors are mostly connected to the vBNS, a high speed ATM research network.

At February 2000 this system consists of 15 machines spread across the USA. Every day each machine takes eight 90 second traces, spread semi randomly across the day.

The first ATM cell of each IP packet is captured, along with a 6 byte 25 MHz timestamp. The ATM cell consists of the 4 byte ATM header, 8 bytes of LLC/SNAP header and 40 bytes of IP. Again for privacy reasons the IP addresses are encoded and any payload present is zeroed out.

The measurement hardware consists of two standard ATM network cards, which provide some limited hardware assistance for timestamping the arrival time of a packet. The clock on the network card has only a very limited period between clock wraps, and the card provides a software interrupt every time this occurs. The measurement software contains an overflow counter from these interrupts. The full packet timestamp is created by concatenating the hardware timestamp with the software counter.

Unfortunately the software interrupt can often be delayed in the interrupt handler, and it is possible that a packet can arrive after the hardware clock has wrapped, but get the old copy of the software timestamp due to the delayed interrupt. This in the most part can be picked up in a post processing phase, as this situation normally causes a packet to seem to have arrived before the previous packet. This case can be tested for and the software timestamp manually adjusted to compensate. On a lightly loaded link, however, it is possible that the error could be missed.

For these reasons, relatively complex and interfering post processing steps are applied to the traces before use, that could influence the results.

The results from the NLANR traces used in this investigation are from a set of traces taken over the 1999, 2000 New Year period, where the duration of traces was increased to 6 minutes, providing longer and more complete VoIP traces, than the usual 90 second traces.

V. VOICE DETECTION AND ANALYSIS

Our analysis of VoIP traffic is restricted to one protocol, H.323, a standard created by the *International Telecommunications Union (ITU)* [4]. The standard, entitled "Packet-based Multimedia Communication Systems", describes a general method for real-time communication over a packet-based network such as IP. The standard allows an H.323 entity to provide real-time audio, visual and data services.

Identifying only one protocol for VoIP may not be as restrictive as could be thought. Since its introduction in 1996 H.323 has become widely supported, with both large software vendors such as Microsoft and hardware vendors such as Cisco, shipping products based around it. With such widespread support, many older proprietary systems are now either including support for, or converting to H.323.

TCP and UDP both use port numbers to identify the application that 'owns' the traffic that arrives. Every time a

TCP or UDP port is requested, a unique port number will be returned. TCP/IP uses a well known port system to provide most services. All of the well known, historic TCP/IP applications, such as HTTP, FTP, SMTP and DNS, have an assigned port number that they will listen on. When one of these services is started, it will request the specific port it should run on, and open that port to listen from. When a client attempts to open a connection to that service, they will request a port to connect out on. This port will be a random unassigned number. Then knowing the address of the machine, and the default port for the service, they can open a connection.

In header trace files, only the source and destination port numbers are available, no payload is present. If one of the ports is a restricted port number it is normal to assume that the data carried in the packet is for that service. If both port numbers are high, as in H.323 no conclusion be drawn as to the type of data carried.

H.323 does not use the privileged range of port numbers. When an H.323 application is started, it requests two specific TCP port numbers to listen on. These are ports 1503 and 1720. These ports are used for call setup and call control, one area of VoIP applications that does need reliable delivery. An H.323 application that wishes to connect to another H.323 user, will connect to that machine on both ports 1503 and 1720. Using these two connections the applications negotiate UDP ports to use for data transfer.

H.323 specifies the use of the *Real Time Protocol (RTP)* [5] for data transfer. This is a general purpose protocol for real-time IP applications. RTP requires the use of two UDP ports, one for data transfer, and one for RTP control information. The data transfer port is defined to be an indeterminate even port number, and the control port number will be the succeeding odd port number. The data port will have large numbers of small, and often fixed sized packets flowing over it. The control port will have much lower data volumes, and will not necessarily have regular arrival times or sizes. Using both this property and the well-known ports, 1503 and 1720, it is possible to attempt to detect H.323.

This project uses a two step process. The first step identifies pairs of IP addresses that communicate on both TCP port 1503 and 1720, and filters the traffic between these two IP addresses into separate trace files. The second step uses human inspection to identify if the UDP data present between these two hosts exhibits any of the indications of RTP data.

Using this system on NZIX resulted in 22 traces identified as possible H.323 connections. The first step identified 71 sessions from the 211 NZIX traces. Of these 71 there were multiple instances of 'port scans' where a host attempts a connection to every port on a destination machine, resulting in connections being attempted on both port 1503 and 1720, which misleads the detection algorithm. The second stage easily rejects these events.

One other circumstance occurred frequently, where a connection was seen on both ports 1503 and 1720, but no more data passed between these hosts. It is possible,

Host 1 Port	Direction	Host 2 Port	Port	Repeat Protocol	Count	Total Length Header Length
0.5.211.254:49606	<-<	0.0.251.56:49608	UDP	C=	4, HL= 5, TL=	88
0.5.211.254:49607	->>	0.0.251.56:49609	UDP	C=	1, HL= 5, TL=	92
0.5.211.254:49606	<-<	0.0.251.56:49608	UDP	C=	6, HL= 5, TL=	88
0.5.211.254:49607	<-<	0.0.251.56:49609	UDP	C=	1, HL= 5, TL=	100
0.5.211.254:49606	<-<	0.0.251.56:49608	UDP	C=	10, HL= 5, TL=	88
0.5.211.254	->>	0.0.251.56	RSVP	C=	1, HL= 6, TL=	160
0.5.211.254	<-<	0.0.251.56	ICMP	C=	1, HL= 5, TL=	60
0.5.211.254:49606	<-<	0.0.251.56:49608	UDP	C=	28, HL= 5, TL=	88
0.5.211.254:49607	<-<	0.0.251.56:49609	UDP	C=	1, HL= 5, TL=	76
0.5.211.254:49607	->>	0.0.251.56:49609	UDP	C=	1, HL= 5, TL=	72
0.5.211.254:1843	<-<	0.0.251.56:1503	TCP	C=	1, HL= 5, TL=	136
0.5.211.254:49606	<-<	0.0.251.56:49608	UDP	C=	10, HL= 5, TL=	88
0.5.211.254:1843	<-<	0.0.251.56:1503	TCP	C=	1, HL= 5, TL=	136
0.5.211.254:49606	<-<	0.0.251.56:49608	UDP	C=	1, HL= 5, TL=	88
0.5.211.254:1843	->>	0.0.251.56:1503	TCP	C=	1, HL= 5, TL=	40
0.5.211.254:49607	<-<	0.0.251.56:49609	UDP	C=	1, HL= 5, TL=	80
0.5.211.254:1843	->>	0.0.251.56:1503	TCP	C=	1, HL= 5, TL=	40
0.5.211.254:49606	->>	0.0.251.56:49608	UDP	C=	2, HL= 5, TL=	88
0.5.211.254:49607	<-<	0.0.251.56:49609	UDP	C=	1, HL= 5, TL=	80
0.5.211.254:49606	->>	0.0.251.56:49608	UDP	C=	1, HL= 5, TL=	88
0.5.211.254:49607	->>	0.0.251.56:49609	UDP	C=	1, HL= 5, TL=	104
0.5.211.254:49606	->>	0.0.251.56:49608	UDP	C=	18, HL= 5, TL=	88
0.5.211.254:49606	<-<	0.0.251.56:49608	UDP	C=	11, HL= 5, TL=	88

Figure 3. H.323 packet flow

although not confirmed, that this could be caused by an attempted H.323 connection that was refused by the destination machine. These were also discarded as not being H.323 connections, as it was the UDP data characteristic that we wanted.

The NLANR data is approximately 7 Gb in size and resulted in 1 Mb of possible H.323 sessions. A large percentage of the traces were uni-directional due to the return route differing from the forward route, and missing the monitoring position.

A. Voice trace characteristics

The measured traces are all similar. To understand the flow of traffic between the two machines, a program was written to print out a text representation of this flow. A sample of this output can be seen in Fig 3. Printed in the first column is an IP address and port on one machine. Next is an indication of the direction of the packet and the other host's IP and port. Then the protocol is listed along with the count field, the header length and total length of the packet in bytes. The count field is the number of identical headers that were seen. This compacts long flows of identical packets to a single line. This occurs on a regular basis with the UDP data channels. It can be seen in this case that both sides are using the same data packet size, 88 bytes. The traffic pattern represented in this output has been chosen as an illustration and contains more control traffic than is usual.

In Fig. 4 the data rates for each of the directions are plotted. These rates only include the UDP traffic seen between these two machines. The UDP control traffic is shown as a separate plot at the bottom of the graph.

The data rates shown on these graphs are achievable by a 33.6kbit per second modem link, a common Internet connection speed in use today. All the graphs consist of a steady base rate between 1kByte per second and 3kBytes per second with excursions about this mean. The amount

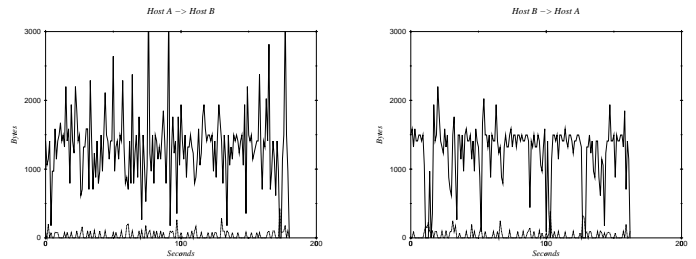


Figure 4. UDP data and control rates

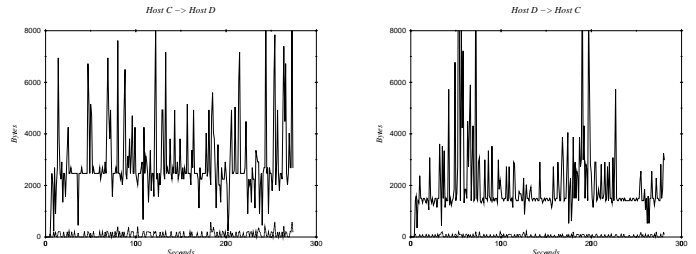


Figure 5. UDP data and control rates

of the excursions and their variability differ between the traces and indeed between the direction of the traces.

Fig. 5 provides another example which shows that both directions need not have the same average bit rate. Although both bit rates are low and well within range of a modem connection.

VI. SIMULATION STUDY

The simulator used in this study is based on a modified version of the ATM-TN simulator[6]. The modifications include the replacement of the ATM transport layer with a simpler bit-serial transport layer and the addition of a UDP protocol model.

ATM-TN's TCP model is retained. This model includes the actual TCP code from 4.4 BSD Lite, modified to suit the simulation environment. Connections are simulated on a packet by packet basis and include slow start, congestion control, fast retransmit, and fast recovery algorithms. [7]

Both HTTP and voice over IP traffic are simulated. The collection of UDP traffic has already been described. Most of the information required to generate the HTTP traffic was obtained from logfiles collected from the New Zealand Internet exchange (NZIX). The trace files used were collected from 3:00pm to 3:10pm each day in July 1997. There were, on average, 421 requests¹ per ten minute interval.

To generate higher loads than those experienced when the trace files were collected, traces for the same time on successive days in July were integrated into a single trace. When even higher loads were required more than one copy of each trace was integrated into the logfile. Each copy was offset in time to minimise the effect of the artificial self correlation of the trace generated in this way.

The TCP MSS and server buffer sizes are not recorded in the HTTP traces we used. To discover these parameters a

¹The complete trace includes requests that were satisfied by the cache or were not successful. There were 421 successful international requests that were not satisfied by the cache hierarchy

connection was established to each host while the network traffic was monitored using tcpdump. From the tcpdump output the MSS and window size was discovered for most hosts. Some hosts did not advertise their MSS. In this case the most common MSS (1480) was used.

The simulation assumes that any routers have sufficient CPU and memory to cause no extra delays and that the delays at the proxy, other than TCP queuing and transmission delays, are negligible.

In many respects this simulation is similar to that described in [8]. The major difference being the addition of the UDP based VoIP traffic.

VII. SIMULATION RESULTS

Figs. 6 and 7 shows the results derived from the simulations and data traces described above. Fig. 6 shows the TCP goodput achieved for different amounts of presented UDP and TCP traffic. The TCP and UDP loads have been normalised so that 1 equals the capacity of the link. The graph shows that for a fixed UDP load the TCP throughput increases linearly with the presented TCP load. However, at approximately the point where the combined UDP and TCP loads reach saturation (1.0) the TCP throughput remains constant independent of how much is presented. Thus the UDP traffic is acting as if it had been allocated a fixed part of the link that the TCP traffic can never use.

The results are displayed in the form of an “impact” plotted against the data rates of UDP and TCP presented to a link. These data rates are normalised to the link data rate which is taken to be 1.0.

Fig. 7 visualises the data in a different way. It shows *impact values* which have been computed by considering the end-to-end rate achieved by a single TCP connection. Firstly, when all the data is TCP and secondly when the data is a mix of TCP and UDP. By taking the ratio of these two rates a number expected to be between 0 and 1 will be obtained. When it is 1 then the inclusion of UDP data has had no additional effect on the throughput. When it tends to 0 then the UDP has displaced all the TCP data.

To understand this graph more fully consider the case when only TCP data is presented to the link (UDP=0). As the presented rate increases from 0 to 1 the throughput of any individual link will remain constant. As the rate approaches 1 and above the achieved rate for each individual connection will start to drop off as they sense congestion and apply congestion control and backoff procedures. This curve of throughput is used as the benchmark that all the other values are compared against.

Now consider the case where there is a mixture of UDP and TCP data. This will be compared with the situation where there is only TCP traffic which is equal to the sum of the presented TCP and UDP rates in the mixed case. The rate of a single TCP connection is found for the mixed case and then divided by the rate for a single connection in the pure TCP case to give the impact value. Thus the impact value measures any *additional* impact of the UDP traffic over the pure TCP case. One consequence is that the impact is exactly 1 for the UDP=0 case as it is

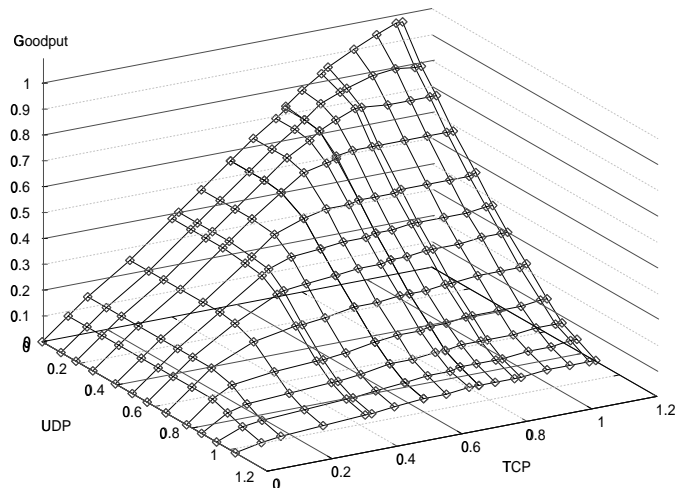


Figure 6. TCP Throughput

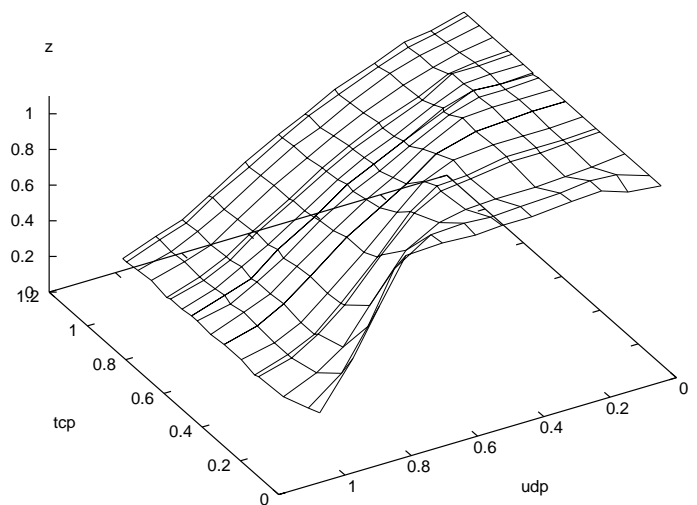


Figure 7. Impact of UDP on TCP

normalised back against itself.

As expected, the impact remains constant at 1 over the region where the total presented load (UDP+TCP) is less than 1. However, as the total presented load rises above this it begins to steeply drop. For example in the region where UDP=1 the impact is approximately 0.2. That is, about 1/5th of the TCP traffic that would otherwise get through will in fact do so.

VIII. CONCLUSIONS

Our original hypothesis is borne out: UDP traffic will have an untoward effect on TCP in congested situations. In cases where there is significant amounts of UDP traffic present it will aggressively displace TCP traffic. In fact the effective available bandwidth seen by the TCP traffic is roughly the link bandwidth less the presented UDP data rate. In the case where TCP only data is present everyone backs off equally but the addition of relatively aggressive UDP data causes a much greater back off in the TCP data. These results clearly have strong implications for traffic

management and the sizing of data links.

IX. ACKNOWLEDGEMENTS

This work is funded in part by NSF Cooperative Agreement No. ANI-9807479.

REFERENCES

- [1] "WAND Group," <http://atm.cs.waikato.ac.nz/wand>, October 1999, Computer Science Department, University of Waikato, New Zealand.
- [2] "University of Waikato Network Overview," <http://operations.waikato.ac.nz/mrtg>, October 1999, ITS, University of Waikato, New Zealand.
- [3] "NLANR Network Analysis Infrastructure," <http://moat.nlanr.net>, October 1999, National Laboratory for Applied Network Research, San Diego, USA.
- [4] "International Telecommunication Union (ITU) Home Page," <http://www.itu.int>, October 1999, International Telecommunication Union.
- [5] H. Schulzrinne, S. Casner, R. Frederic, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," *Internet RFC 1889*, January 1996.
- [6] M. Arlitt, Y. Chen, and R. Gurski, "Traffic modeling in the ATM-TN TeleSim project: Design, implementation, and performance evaluation.," in *Proceedings of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, July 1995.
- [7] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *Internet RFC 2001*, January 1997.
- [8] Murray Pearson and Anthony McGregor, "a simulation study of network architectures to support http traffic on symmetric high-bandwidth*delay circuits "," in *Proceedings of the Asia-Pacific Web Conference (APWEB'99)*, 1999.

