A Connectionless Approach to Providing QoS in IP Networks

B. Nandy, N. Seddigh, A.S.J. Chapman and J. Hadi Salim
Computing Technology Lab, Nortel,
PO Box 3511, Station C
Ottawa, ON K1Y 4H7, Canada
E-mail: {bnandy, nseddigh, achapman, hadi}@nortel.ca

Abstract: The attempt to provide QoS in IP networks has raised some interesting questions on how a service can be provided to meet the application requirements while obeying the network resource constraints. Previous efforts focussed on a flow-based, connection oriented approach to deliver QoS for IP Networks - Intserv. This approach was quite comprehensive but it has not been widely deployed because of complexity and scalability issues. A recent packet marking based scheme called Differentiated Services (Diffserv) Architecture provides a relatively simple and coarse approach. It is too early to predict the usefulness of this approach. This paper outlines a framework to deliver IP QoS which is based on Intserv. It addresses scalability concerns by removing the need for a connection-oriented reservation setup mechanism and replaces it with a Diffserv-like mechanism to consistently allocate bandwidth end-to-end in a network. A prototype device is discussed that manages bandwidth on a node. An algorithm is presented that allows the device to automatically detect application QoS requirements without the need for application-level signalling. A priority-based scheduling mechanism with a variant of weighted round-robin is described.

Keywords: QoS, IP Networks, Connectionless, Diffserv, Intserv

1.0 Introduction

The Internet has traditionally offered services in a best-effort manner. This is acceptable in an environment where congestion due to bandwidth requirement is seldom. Moreover, most of the traditional applications like email, file transfer, news are relatively insensitive to delay and delay variations. However, recent surge in Internet popularity has caused the bandwidth to be at premium. The new applications like streaming video have high bandwidth requirements as well as delay constraints. Web access, real-audio etc. require a service which is better than best-effort. Adding more bandwidth is no longer a solution. Thus, a mechanism is needed for bandwidth sharing and prioritization. The idea of prioritizing traffic is also in tune with the approach of commercializing internet applications i.e., to get better service, one needs to pay more.

A mechanism for bandwidth management is needed by which the finite available network resources can be shared among various applications in a manner suitable to the specific applications and also following the guidelines from the administrator.

1.1 Providing QoS via Integrated Services (Intserv) Architecture

Initial efforts on providing QoS for IP networks focussed on the Intserv (Integrated Services) model. This model relied on a flow-based, connection-oriented approach to deliver QoS for IP networks. An overview of Intserv is provided in RFC 1633 [1]. In this RFC, Intserv extends the current IP architecture so that it provides QoS to end users. The extension includes: (i) A service model with two services (ii) A reference implementation framework to provide support for QoS-enabled routers.

The network element behaviour required to support the two services are outlined in: (i) RFC 2211: Controlled Load [2] and (ii) RFC 2212: Guaranteed Service [3]. The reference implementation framework provides implementation-level detail to realize the above two services.

The framework [1] proposed to provide QoS support in routers includes the following four elements: (i) classifier, (ii) scheduler, (iii) admission controller, (iv) reservation setup protocol. RSVP is used as the reservation setup protocol of choice [4].

1.2 Providing QoS via Differentiated Services (Diffserv) Architecture

The Intserv model is quite mature and much work has been completed. However, it has not been widely deployed due to a variety of concerns. A primary issue is that of scalability [6]. With Intserv, intermediate routers need to save per-flow state information. Another concern is the end-to-end connection-oriented approach of RSVP [4] which is foreign to IP networks. The above issues result in an architecture that is complex to implement and deploy. This might be the single most important reason why the new Differentiated Services initiative has started.

The Differentiated Services Architecture (Diffserv) attempts to address the above concerns by operating on the premise that it is beneficial to move complexity to the edge of the network and keep the core simple [12]. It intends to provide differing levels of service in the Internet without the need for per-flow state and signaling at each router. The framework includes the following elements: (i) A Traffic Conditioning element at the edge of the network that performs the following: marks packets to receive certain levels of service at the backbone, policies packets and performs traffic shaping to ensure that packets entering the backbone conform to network policies (ii) Core routers that treat packets differently depending on the packet marking completed by the edge device (iii) Allocation/policy mechanism that translates into end-to-end QoS levels of service seen by the end-users [12].

1.3 Our Work

This paper discusses a scheme for providing end-to-end QoS which is driven by application requirements. The approach is based on Intserv but does not utilize RSVP, thus addressing the scalability concerns [6] expressed about the Intserv model. Our approach consists of the following: The first element is a device called the Traffic Conditioner which manages the bandwidth at router junction points. The second element is a connectionless mechanism for ensuring consistent end-to-end delivery of QoS based on application requirements.

The Traffic Conditioner is based on the reference implementation framework described in RFC 1633. It contains three of the elements required to manage bandwidth on a particular router junction point: (i) Classifier (ii) Admission Controller (iii) Scheduler. However, instead of RSVP, it utilizes a scheme proposed in [5] to automatically discover QoS requirements for traffic flows and services them accordingly.

In the RSVP model, host machines would initiate connection-setup end-to-end before engaging in data transfer. This connection-setup is used to communicate application QoS requirements and determine if the necessary network resources are available before starting data transfer. This type of pre-negotiation is not intrinsic to IP networks.

The Traffic Conditioner automatically and dynamically meets application requirements without the need for pre-negotiation. Instead, it performs on-the-fly traffic characterization to put network traffic into different classes which are then serviced by a scheduler in such manner that reflects application requirements for that class of traffic.

Having discovered the requirements for a particular flow, and classified the packet accordingly, the Traffic Conditioner marks the packet with its class.

The second element required is a mechanism to ensure consistent allocation of bandwidth across all routers to provide end-to-end QoS. This is an area of ongoing research and is discussed further in Section 3.0. The connectionless approach utilized is similar to the scheme used in the Differentiated Services Architecture outlined earlier.

The paper is organized as follows: Section 2 describes the functional detail of the Traffic Conditioner including the classification, admission control and scheduling schemes utilized. Section 3 outlines the necessity of consistent bandwidth allocation mechanism. Section 4 presents the experimental results gained from implementing the prototypical Traffic Conditioner. Section 5 contains the conclusion and discusses future plans.

2.0 Traffic Conditioner: Functional Detail

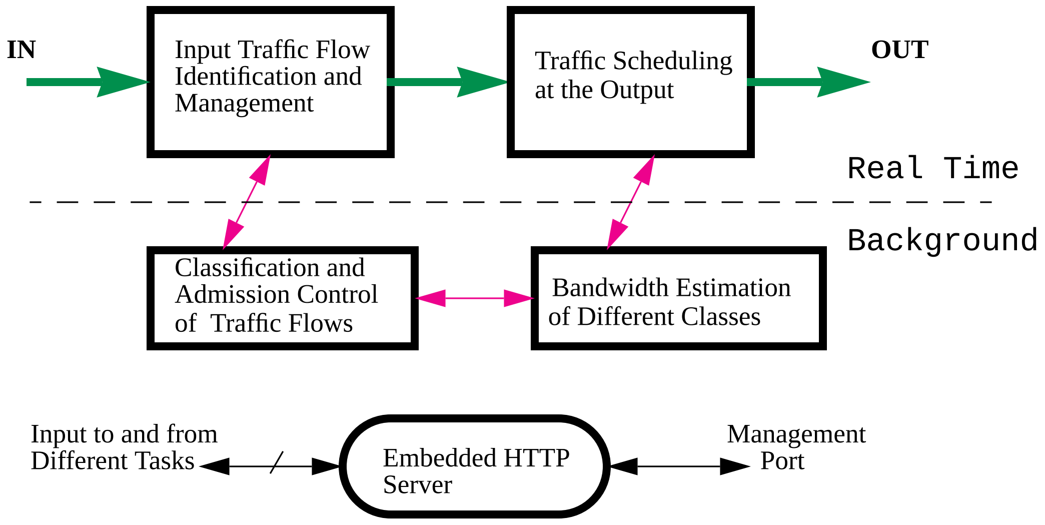
With the advent of new applications, the traffic pattern in IP networks (Internet) has changed over the years. The present traffic at the IP networks can be broadly divided into three categories in: (i) Real-Time, (ii) Interactive and (iii) Bulk. The real time voice and video requires an uninterrupted data stream which keeps the maximum delay variation within a limit. Video conferencing requires the total delay to be within a limit so that the human interaction is not affected. The traffic generated from Telnet, X-windows and web browsing are also interactive in nature and requires a good response time. Another category is bulk transfer like file transfer and NFS backup which do not have any stringent delay variation requirement. At the same time, this bulk traffic should not cause congestion for other classes of traffics.

Figure 1 shows a Traffic Conditioner functional model. Traffic Conditioner performs bandwidth management on an individual packet stream (i.e., the packets flowing between two applications in client and server). Each packet belongs to a flow. Flows are uniquely identified by source and destination ip addresses, transport level port pairs and protocol type. Flows are important, since the classification is performed by the characteristics of a stream of packets between a pair of users.

Any packet arriving at the input of the Traffic Conditioner is associated with a flow. All the flows are maintained in a flow-list. If the arrived packet does not belong to any existing flow in the flow-list, a new flow entry is attached to the flow-list. The packet is queued at the scheduling class queue for the flow's class. The scheduler outputs the packets based on the class and the scheduling strategy. The real-time data path has two major functions: (i) Identify the flow for the input packet and (ii) schedule the packet to the output. A flow entry to a flow-list is removed if there is no packet arrival to the flow for a fixed time (e.g., two seconds).

The background functions are to (i) classify the flows, (ii) admission control and (iii) perform bandwidth estimation of different classes of traffic. The classification of each flow is performed on the fly based on traffic characteristics (e.g., bit rate, packets per second etc.). The classification is performed periodically on all the flows in the flow-list. The packets arriving at the input before the proper flow classification are scheduled with default class.

FIGURE 1. Traffic Conditioner Functions


IN
Input Traffic Flow Identification and Management
Traffic Scheduling at the Output
OUT
Real Time
Background
Classification and Admission Control of Traffic Flows
Bandwidth Estimation of Different Classes
Input to and from Different Tasks
Embedded HTTP Server
Management Port

The bandwidth estimator updates the bandwidth usage on each class periodically. This is necessary for the admission controller and the classifier at the classification time. An embedded HTTP server is provided for monitoring and setting parameters by the administrator.

2.1 Flow Classification and Admission Control

The paper [11] proposes a hierarchical link sharing mechanism to address the requirements for realtime traffic in presence of other classes of traffic. The classes in the hierarchical link-sharing structure can be multiple agencies, multiple applications, multiple protocols etc. and the mechanism used for link sharing is called class based queueing (CBQ). The experimental results reported in [9], show that the CBQ based link sharing mechanism between CBR type of traffic with two different priority of TCP traffics. It also shows the delay behavior of the traffics in the shared link.

The Traffic Conditioner classification strategy is based on the scheme proposed by [5]. The flow classification for both TCP and UDP flows are performed on the basis of different treatments required for different traffic types i.e., the application requirements. Thus, different applications with the similar service requirement will fall under the same class. These classes can be different nodes in the hierarchical link-sharing structure as proposed in [11].

The idea behind classifying TCP flows is to separate traffic requiring fast response from the delay insensitive bulk transfer. The UDP flows are classified to differentiate between traffic requiring (i) low latency and low bandwidth, (ii) low latency and high bandwidth and (iii) delay insensitive bulk transfer. The classification of a traffic flow is performed on the basis of traffic characteristics rather than identifying the well known ports (e.g, 80 for web) although that can be used to assist the classification.

The Traffic Conditioner divides traffic into the following classes:

Interactive: The TCP flows with short packets and requiring short round trip time are captured in this class. The applications like, Telnet, web browsing and interactive X-windows etc. will fall in this category. The objective here is to protect a portion of the total bandwidth to ensure a reasonable response time. A short packet is defined as a packet with less than or equal to 128 bytes.

All the TCP flows are classified as interactive at the beginning. If the number of continuous long packets exceed a threshold (e.g., 200) without a string of two or more short packets, the flow is moved to bulk transfer class. If the class bandwidth is available, the class is considered as Bulk Transfer with Reserved bandwidth. Otherwise, the flow class is Bulk Transfer with Best Effort.

Bulk Transfer with Reserved Bandwidth: The TCP flows with continuous long packets are captured in this class. Applications like, large FTP, Web image transfer will fall in this class. It is ensured that the bulk flows get a certain portion of the total bandwidth and at the same time bulk traffic should not encroach in allocated bandwidth of other classes of traffic.

If there are two or more continuous short packets in the stream of long packets, the flow is reverted back to Interactive class.

Bulk Transfer with Best Effort: The TCP flows with continuous long packets but no class bandwidth available at the reserved category falls in this class. This traffic is scheduled on best effort basis.

If bandwidth becomes available, the flow makes a transition to Bulk Transfer with Reserved Bandwidth class. If there are two or more continuous short packets in the stream of long packets, the flow is reverted back to Interactive class.

Low Latency: The UDP flows with low packet rates are captured in this class. The applications like real audio, interactive voice, NFS requests and short replies, DNS transactions fall in this category. The objective of this class of traffic is to treat it with high priority so that the latency remains low.

All the new UDP flows are classified as Low Latency at the beginning. The flow is moved to UDP Best Effort if the packet rate exceeds packet rate threshold or no bandwidth is available at Low Latency class.

Best Effort: The UDP flows with high packet rate but not classified as Real Time class falls in this class. NFS file backup is one application belongs to this category. The traffic is transferred in a best effort basis.

The flows matching the Real Time template are moved to the Real Time class if that class has available remaining bandwidth. Flows can also have a path back to Low Latency class.

Real Time: The UDP flows like streaming video and NFS based video are in this class. The traffic is handled with high priority so that the latency remains at the minimum. The reason for distinguishing Low Latency and Real Time is to preserve the bandwidth for Low Latency class of traffic.

The streaming real time traffic arrives at a constant rate (with a slight variation due to network delay) at the receiver. The distribution of packet interarrival times is uni-modal. On the contrary, NFS based file transfer, sends a fixed number of packets before it waits for acknowledgment from the application layer. It means that the distribution of packet interarrival time for NFS based application is bi-modal which is different from unimodal distribution of Real Time class of traffic. Also, if the packet rate of the flow in Real Time class exceeds a certain threshold (which makes it as unlikely to be video), the flow is reverted back to UDP Best Effort class. If the flow is idle for more than a second, its class is changed to Low Latency.

As mentioned in [10] to provide bounded delay service, networks must use admission control to regulate the load. In this work, the measurement based admission controller enforces a limit on the flows which require high bandwidth and high priority scheduling. The admission control is enforced on the Real-time flows based on the measurement of link usage. Bandwidth Estimator periodically updates the bandwidth usage of Real-time class. Any new Real-time flow is admitted (i.e., allowed to be serviced) only if the new bandwidth requirement of the class is within the administrator specified limit. Otherwise, the flow is marked as *reject.* Packets of flows belonging to reject class are dropped by the traffic conditioner. An ICMP host unreachable message is sent back to the source host to stop the flow.

2.2 Scheduling

A key component in bandwidth management of a link is a scheduling mechanism. In the work [11], the usefulness of priority scheduling mechanism for link sharing is studied. The scheduling of the classified flows in Traffic Conditioner is performed with two priorities: high and low. The high priority traffic is scheduled without any delay at the scheduling queue and limited by the admission control mechanism. The low priority traffic is scheduled according to a set of rules based on the allocation of bandwidth and a criteria for sharing bandwidth with other classes of traffic on the link. Traffic in the Real Time and Low Latency classes are handled with high priority. Traffic in the Interactive, Bulk Transfer with Reserved Bandwidth and Best Effort classes are scheduled with a low priority. The ideas are: to protect a portion of bandwidth for Interactive class of traffic to guarantee a low round trip time, to limit the delay insensitive Bulk Transfer traffic to a specified limit so that it does not hog bandwidth from other classes. The rule based approach for scheduling of classes with low priority will achieve a similar goal of weighted round robin [11], with weights proportional to the combination of allocated bandwidth per class and delay sensitivity of the class.

A scheduling window of T seconds (1 sec. for this implementation) is chosen. The bandwidth allocation for each class during T seconds is proportional to the administrator specified bandwidth values. The window is divided to sub-windows of a smaller t milli seconds intervals. The scheduler wakes up on every t seconds and schedules the packets arrived at different class queues during t seconds with the following rules:

1. All the high priority packets (UDP Low Latency and Real Time) are transmitted until there are no packets remaining in these queues. The queues are served in a round robin fashion.
2. The packets at Interactive and Reserved Bandwidth classes are transmitted only if there is no remaining high-priority packet at the class queue. These two queues are served in a round robin fashion.
3. The packets from Reserved Bandwidth class queue are transmitted only if the allocated bandwidth limitation for the class during scheduling window of T seconds is not exceeded.
4. The packets from Interactive class queue are transmitted even if the allocated bandwidth for the class is exceeded (in a scheduling window of T seconds) but there is no packet waiting at the Best Effort and Reserved Bandwidth classes and bandwidth is available in those two classes.
5. The best-effort class of packets are transmitted if there is no packet in any other class of traffic and bandwidth is available for this class. These packets are also transmitted if its allocated bandwidth is exceeded (in a scheduling window of T seconds) but there are no remaining packets at the Reserved Bandwidth class and bandwidth is available for this class.
6. Packets from Reserved Bandwidth class and Best Effort class are not allowed to borrow bandwidth from Interactive class of traffic. Thus, the interactive bandwidth is preserved.

The scheduling class queue length at the traffic conditioner for different classes of traffic are different. The queue length will never grow for high priority traffic. The queue length does not grow for interactive traffic, since the bandwidth is preserved for this class and allowed to steal bandwidth from other classes. Queue length of Bulk Transfer with Reserved Bandwidth and TCP Best Effort will grow with traffic but TCP adjusts to keep the queue length minimal. The queue length of UDP Best Effort has the possibility of growing if the traffic exceeds its allowed limit. Thus a congestion control mechanism similar to drop tail is implemented to restrict the excessive traffic for this class.

3.0 Connectionless Approach for End-to-End Bandwidth Allocation

The Traffic Conditioner is one element in the connectionless approach to QoS in IP networks. Ongoing investigation and experimentation is being pursued to study the second component. i.e. providing a mechanism to emulate end-to-end reservation setup. The issue of providing QoS similar to Intserv's Guaranteed Service Model [3] without using an RSVP-like resource reservation mechanism, remains an open research area. It is not clear whether or not the Differentiated Services initiative will be able to provide this service.

However, we believe that it should be possible to achieve the Controlled Load Service model [2] using the approach outlined in this paper. In order to do this a mechanism is required to treat flows in a consistent manner in all routers along the path from source to destination. To avoid the pitfalls of RSVP, this should be achieved without per-flow state saving at each router.

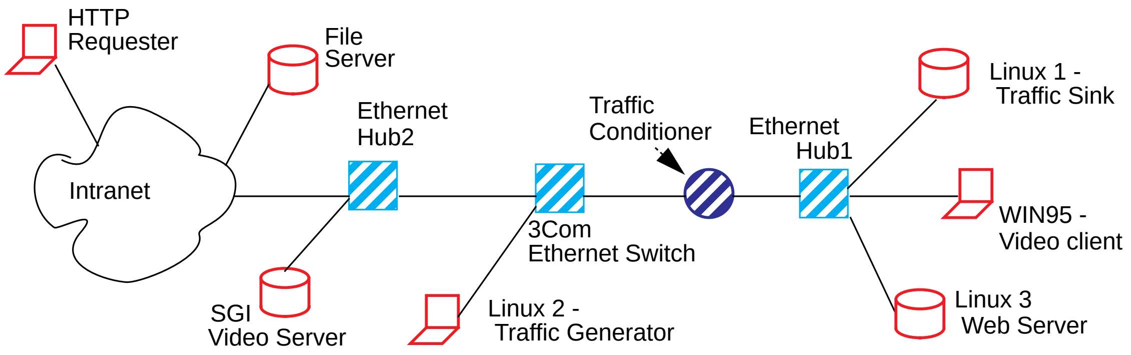
This can be achieved in a well-engineered network with packet marking by the Traffic Conditioner. The administrator statically preallocates bandwidth for each of the defined classes in a consistent manner across the network. This can easily be achieved in an intranet environment. However, in the internet, the solution evolving out of the Diffserv initiative should be applicable. Proposals have been made that discuss dynamic allocation of bandwidth in a connectionless environment [7].

4.0 Experimental Results

The Traffic Conditioner (TC) was implemented on a Pentium 200MHz PC running VxWorks as the RTOS. The current implementation utilizes a 4-port OSICOM PCI ethernet card. One port is used for network management. The other two ports connect to the link that the Traffic Conditioner is conditioning. The TC also contains Mombasa - an embedded web server that is used to facilitate device network management. With the inclusion of Mombasa, network administrators are able to manage the Traffic Conditioner using any standard HTTP web browser. e.g. Netscape or Internet Explorer. Currently, Mombasa is used for configuration and statistics monitoring.

Although this study would have benefitted from a hardware implementation, the software implementation of the TC was able to support traffic levels on a 10Base-T ethernet LAN.

FIGURE 2. Experimental Setup



For its operation, the TC sets the ethernet cards in promiscuous and NSAI modes. Promiscuous mode allows it to take in all packets on the line. NSAI mode prevents the ethernet card from stamping its MAC address as the source. This allows the TC to operate transparently on the link.

The implementation model used by the TC closely reflects that depicted in Figure 1 with the Flow management, and Scheduling processes carried out as foreground processes and the classification carried out as a background process.

As mentioned previously, RFC 1633 [1] provides a reference implementation model for a QoS-enabled router. The intention of this set of experiments was to study the QoS model elements referred to in that RFC. To do this, the TC was developed as a stand-alone device intended to perform bandwidth management over a single link. The rest of this section describes the results obtained when the Traffic Conditioner was deployed on the network in the presence of various types and rates of traffic.

The experimental setup is depicted in Figure 2. As the Figure shows, the TC was deployed in a live network. To carry out the tests, varying levels of network traffic needed to be created. A deterministic traffic generator was used to generate traffic with varying packet lengths and sending rates for both UDP and TCP. As the Figure 2 shows, the setup consisted of an SGI video server, a File Server, a Windows 95 computer with streaming video client software, Linux2 which served as the Traffic Generating computer, Linux1 which served as the sink computer for the traffic generator, and Linux 3 which was used as Web server.

A key outcome of the experiments below was the verification of the class scheme proposed in [4]. Various types and rates of traffic were generated and it was observed that all fit into expected classes. The experiment was performed with the following applications: ftp, telnet, streaming video, real audio, web browsing (small and large files), DNS and NFS based file transfer.

FIGURE 3. Pkt Interarrival Times for Streaming Video - without TC; Background Load - 0.5Mbps

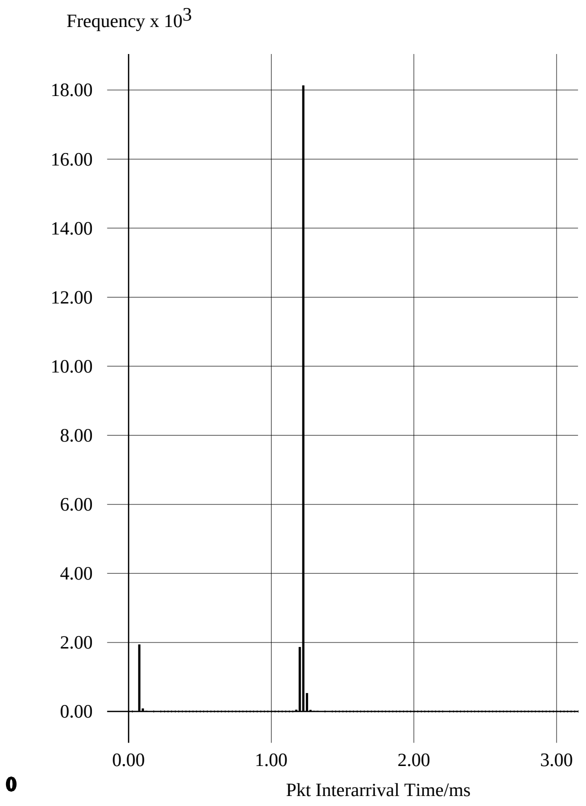
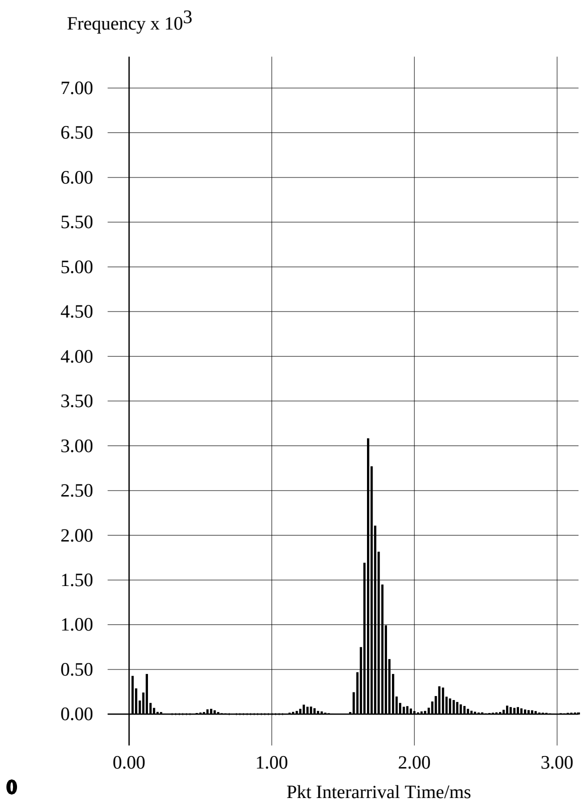


FIGURE 4. Pkt Interarrival Times for Streaming Video - without TC; Background Load - 6.2 Mbps



Experiment 1 - Behavior of Video Traffic in presence of network traffic

The goal of the 1st experiment was to explore how real-time traffic such as video behaved in the presence of network traffic. It was desirable to observe its behavior in scenarios with and without the TC. Video sessions were started from the Win95 computer. This resulted in around 1.4 Mbps of streaming video being transferred from the SGI video server to the Win 95 video client. Background network traffic was generated with Linux2 as the source and Linux1 as the sink.

Three separate tests were carried out. Packet interarrival times were measured for the video stream in the following test cases:

1. Without Traffic Conditioner; Low network traffic rates
2. Without Traffic Conditioner; High network traffic rates
3. With Traffic Conditioner; High network traffic rates

In all cases, the experiment was run for 3 minutes and around 25,000 samples obtained. Measurements were obtained on the TC interface connected to Ethernet Hub1. The results of the three tests can be seen in Figures 3, 4 and 5. Each Figure plots a histogram of the packet interarrival times for the video traffic.

FIGURE 5. Interarrival Times for Streaming Video - with TC; Background Load - 6.2Mb

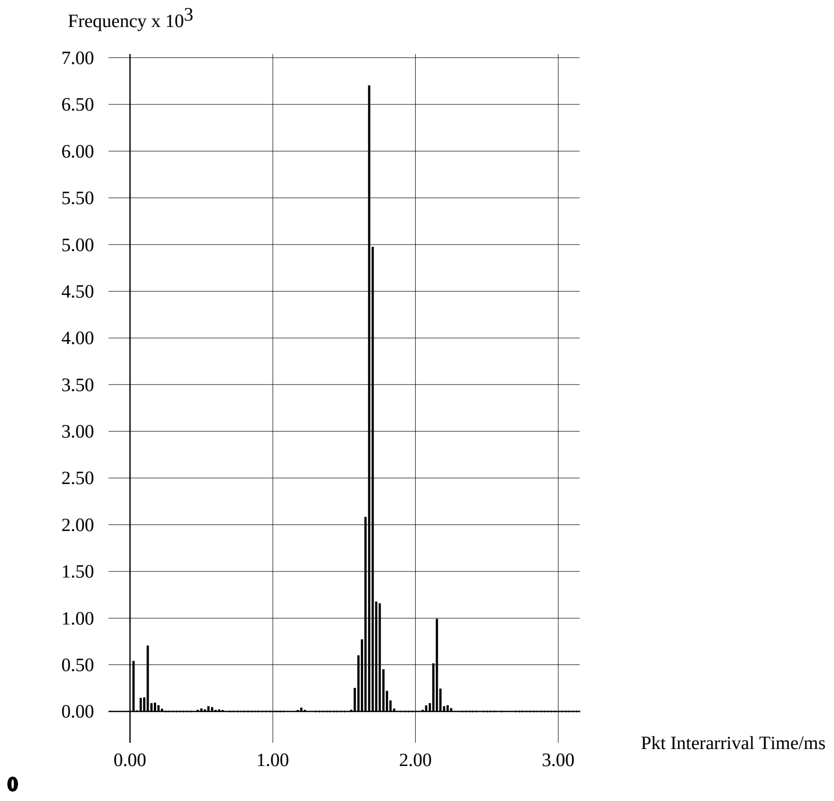
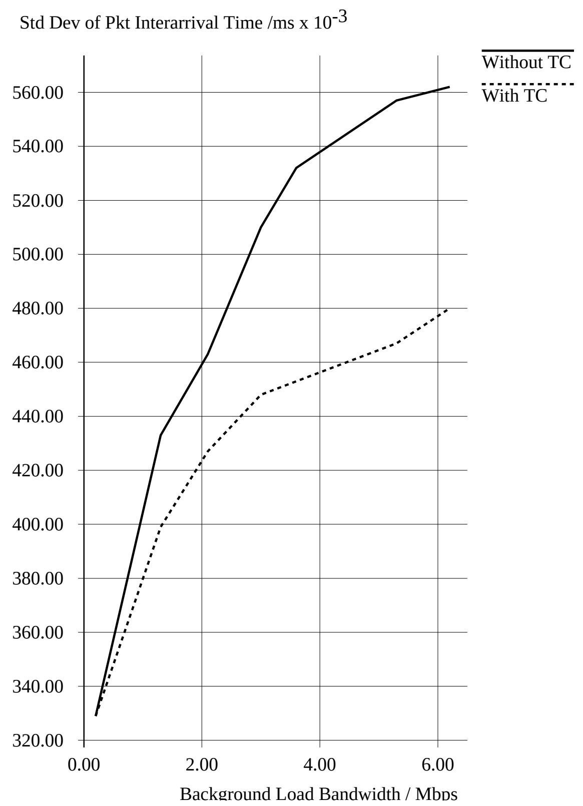


Figure 3 shows the packet interarrival distribution for a network with very low traffic levels. The distribution is a clear reflection of video traffic characteristics. Two clear peaks can be seen on the distribution. There was a third peak at around 80 ms but that set of data was filtered out for all three Figures as it is not much of comparative value. The peak around 1.25 ms accounted for almost 84% of the 25,000 data samples. The other peaks at 0.1 and 80 ms appeared equal in height and width. This reflects the nature of traffic from this particular video stream which we verified sends packets in 12-packet cycles - 10 packets with a delay of 1.25 ms, 1 packet with a delay of 0.1ms and 1 packet with a delay of 80 ms.

The next testcase involved the same video session but also, introduced a background network traffic load of 6.2 Mbps from the Linux2 to the Linux1 computer. The measurements from this test can be seen in Figure 4. As can be seen from the Figure 4, the height of the main peak reduced greatly and its width expanded considerably. Quantitatively, the width expanded from 0.23 ms to 0.45 ms, almost doubling in size. This kind of effect on video traffic is not desirable as it affects the quality of images that is received by the client. In addition, four small peaks appear due to the contention of background and video traffic at the physical layer (ethernet).

FIGURE 6. Standard Deviation of Packet Interarrival Times for streaming Video



The final testcase in this experiment involved the deployment of the TC (as shown in Figure 2 on page 6) and the launching of a video stream in the presence of the same 6.2 Mbps background network traffic. The results of the measurements are displayed in Figure 5. As can be seen, the main peak has a much higher height than that of testcase 2 (Figure 4). Also the width of the peak has reduced from 0.45ms to 0.3ms - a reduction of 33%. Secondly, the two peaks on the right side of the main peak have been consolidated into a single small peak with minimal width. The TC is unable to remove this peak because it does not have control over the delays that occur as a result of high collision rates between the video and background traffic at the physical layer. On a final note, the small peaks to the left of the main peak have virtually been eliminated. The results of Figure 5 are quite encouraging as they reveal the TC's ability to improve the quality for a video stream in the presence of heavy network traffic.

Figure 6 shows the comparison of the interarrival standard deviations for different levels of network load with and without the TC deployed. As can be seen from Figure 6, the two lines diverge with increasing background load. As expected, the TC appears to be showing more value as the background traffic load increases.

Experiment 2 - Behavior of TCP Bulk Transfer with Reserved Bandwidth

The goal of the next experiment was to observe the effects of bandwidth clamping on TCP Bulk transfer sessions. FTP sessions were used to setup TCP connections between Linux 1 and file server (see Figure 2). Without the traffic conditioner, it was seen that transfer of large files consumed average bandwidth of 488 Kbps.

The same test was repeated with the Traffic Conditioner deployed. The administrator specified rate for the TCP Guaranteed class of traffic was set to 300 Kbps. The file transfer continued to attempt transferring files as fast as it could. For the first few seconds, the data transfer rate for the flow remained at 488 Kbps. However, the mechanism of the Traffic Conditioner based on queueing delays worked to effect end-to-end TCP rate control.

The result being that the application adjusted its sending rate of data to the amount that the Traffic Conditioner would allow through the pipe. The administrator specified rate was 300 Kbps and measurements at the Traffic Conditioner output showed that the Traffic Conditioner could clamp the flow down to 292 Kbps - to an accuracy of 97% percent. The maximum queue length for this class was observed to be 11.

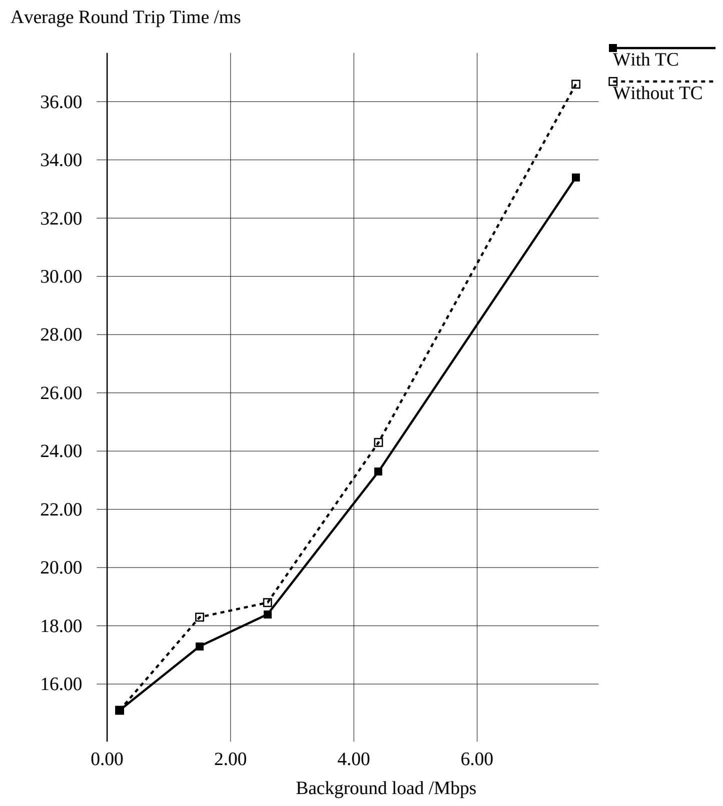
The experiment was also performed with two ftp sessions attempting to transfer large files. It was observed that the two flows shared the available bandwidth of 300 Kbps for this class. One flow was transferred at 130 Kbps and the other at 166 Kbps. The maximum queue length was observed to be 21 for this class of traffic.

Experiment 3 - Behavior of TCP Interactive in the presence of network traffic

The goal of this experiment was to see whether or not the round trip time of TCP Interactive class of traffic suffered in the presence of background traffic. A software tool (HTTP requester) was used to generate HTTP requests to retrieve a file from a web server. The web server runs on the Linux 3 (Figure 2) machine. Video was played at the background to generate high priority traffic. Also, traffic of UDP Best Effort class is generated as background traffic.

The round-trip time for each HTTP request was measured to represent a quantitative indicator of user-experienced delay when using a web browser. The measurement was performed under varying network load. Each measurement is the average of 100 HTTP requests' round trip time. Figure 7 shows the round trip times as a function of network load. It is observed that the RTT is lower when TC is deployed. This is due to the fact that a certain bandwidth is preserved for the Interactive class of traffic. The impact of TC is more prominent at high background load. However, it is debatable if a reduction of 2 mili-seconds of RTT has any impact on the user perception of web browsing.

FIGURE 7. Average Round Trip Time for TCP Interactive Traffic



5.0 Conclusion and Future Work

The key contribution of this work is to show that it is possible to automatically discover the application QoS requirements without RSVP-like pre-negotiation. Further, experimental results have shown that the on-the-fly classification scheme proposed in [5] can successfully classify the current traffic in IP networks. This implementation has also demonstrated that the combination of classifier, scheduler and admission controller can effectively condition and manage traffic on a link. It has been shown that the different classes of traffic can be serviced in a "required" manner in the presence of background load.

The Traffic Conditioner is one building block in providing QoS capability in IP networks. Investigation and experimentation is needed for other building blocks. The other key building block is a mechanism for providing end-to-end allocation of service without a per-flow connection-setup protocol. This is an area of on-going work.

One open issue is to understand if the flow-based connectionless approach outlined in this paper is scalable. Although this approach removes the scalability issue associated with per-flow state saving at routers, there is still an issue with the number of flows that need to be handled by Traffic Conditioning elements. A second area of exploratory work is to investigate the suitability of the Traffic Conditioner as an edge device [8] in a Differentiated Services network [12].

6.0 Acknowledgments

We would like to thank Steve Jaworski for making different software tools available and Joseph Thoo for discussion during various phases of this work.

7.0 References:

[1.] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", June 1994, Request for Comments: 1633

[2.] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", September 1997, Request For Comments: 2211

[3.] S. Shenker, C. Partridge, R. Guerin, "Specification of Guaranteed Quality of Service", September 1997, Request for Comments: 2212

[4.] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", September 1997, Request for Comments: 2205

[5.] A. Chapman and H. T. Kung, "Automatic Quality of Service in IP Networks", Proceeding of the Canadian Conference on Broadband Research, Ottawa, April 1997, pp. 184-189

[6.] R. Guerin, S. Herzog and S. Blake, "Aggregating RSVP based QoS Requests", Internet Draft, November 1997

[7.] K. Nicholas, V. Jacobson and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", Internet Draft, draft-nichols-diff-svc-arch-00.txt, November 1977

[8.] D. Clark, and J. Wroclawski,"An Approach to Service Allocation in the Internet", Internet Draft, draft-clark-diff-svc-alloc-00.txt, July 1997

[9.] I. Wakeman, A. Ghosh, J. Crowcroft, V. Jacobson and S. Floyd, "Implementing Real Time Packet Forwarding Policies using Streams", Usenix 1995 Technical Conference, January 1995, New Orleans, Lousiana, pp. 71-82.

[10.] S. Jamin, P. B. Danzig, S. J. Shenker and L. Zhang, "A Measurement-based Admission Control Algorithm for Integrated Services Packet Network", Proc. of ACM SIGCOMM'95, pages 2-13, 1995.

[11.] S. Floyd and V. Jacobson, "Link-sharing and Resource management Models for Packet Networks", IEEE/ACM Transactions on Networking, Vol. 3, No. 4, August 1995

[12.] Nichols, K. and Blake, S., "Differentiated Services Operational Model and Definitions", Internet Draft, draft-nichols-dsopdef-00.txt, February 1998