

An Architecture for Automated Network Control of QoS over Consumer Broadband Links

Lawrence Stewart, Grenville Armitage, Philip Branch, Sebastian Zander

Centre for Advanced Internet Architectures, Swinburne University of Technology

{lastewart,garmitage,pbranch,szander}@swin.edu.au

Abstract—This paper proposes a novel architecture for meeting Quality of Service (QoS) requirements of real-time traffic across consumer broadband links. In our approach the responsibility of QoS signalling is moved away from the application to the network. Network servers automatically identify traffic that might benefit from QoS and then trigger the provisioning of QoS by signalling network elements such as access routers. This approach removes the need for the application to signal to the network its explicit QoS requirements, making applications easier to develop and more portable. It also enables QoS provision for legacy applications for which there is limited opportunity to include explicit end-host signalling protocols. The paper develops the architecture required to realize the approach and discusses the underlying techniques.

Index Terms—Communication Systems, Network Traffic QoS Control, Traffic Classification

I. INTRODUCTION

Quality of service (QoS) provisioning across the Internet has been a challenging area of research for over a decade. It has resulted in a number of standards and approaches, such as Integrated Services (IntServ) [1], the Resource Reservation Protocol (RSVP) [2] and Differentiated Services (DiffServ) [3]. Yet in terms of the number of networks and applications that use these technologies, success has been modest at best. Most networks and applications continue to exclusively use ‘best effort services’. Usually they do not expect, and they are not given, explicit QoS guarantees.

One of the reasons for the poor uptake of these QoS approaches is the requirement for applications to signal explicit quality of service requirements to the underlying network. This is an onerous obligation to place on applications and their developers. It presupposes that the developers are aware of the issue, understand the technologies for providing QoS and have explicit network requirements for their application. None of these requirements are necessarily true.

Software developers often do not have a good understanding of network traffic engineering. Expecting them to understand the subtleties of different QoS classes is unreasonable. Often, where there is an understanding of the issues, their expectations of the network are quite vague.

While some applications (such as Voice over IP) with a long history of development, might have well-understood and explicit QoS requirements, most emerging applications, such as multi-player games, do not. QoS requirements for emerging applications are often quite difficult to state explicitly. Applications where quick responses are essential (such as First Person Shooter games) may simply require that all information

is propagated as quickly as possible. Other applications may require that some information needs to be distributed rapidly, while other information can be distributed much more slowly. Specifying such vague requirements in explicit terms suitable for implementation as (for example) RSVP protocol exchanges is likely to be challenging for most developers.

Then there is the issue that most network operators do not currently support mechanisms for the dynamic provisioning of QoS for certain applications. The main reason is that applications do not support QoS signalling. Although basic mechanisms for traffic differentiation exist in many current routers, any solutions based on complicated signalling protocols are far more complex to manage than best effort networks. Therefore even where the QoS requirements of an application are well understood, developers will not implement QoS support as long as the underlying networks do not support it and there is no simple and standardized interface/protocol. Restricting an application to run exclusively in an IntServ or DiffServ environment will drastically reduce the number of potential users. The alternative of implementing all the protocols that might be used in networks requires a large investment in software development, as well as a profound understanding of networks, QoS protocols and architectures.

Consequently, if QoS is to be made available to new applications we must find alternatives to explicit QoS signalling from the application to the network. Furthermore, QoS should also be made available to existing applications without the need to change their implementation.

In this paper we propose a novel architecture for moving QoS signalling from the application to the network. We propose placing intelligence within the network to identify traffic flows that might benefit from QoS. We identify the major functions needed in such an approach and suggest what we believe are appropriate groupings of functions.

The rest of the paper is structured as follows. Section II discusses related work. In particular we describe some techniques and products that have been developed recently which might form the building blocks for an integrated architecture. Section III outlines our proposed architecture: the Traffic Classification and Prioritisation System (TCAPS). Section III.C describes the functional groupings that we believe are most appropriate. Section V describes our prototype design, including a detailed description of all components of the architecture. Section VI concludes with a discussion of research issues that need resolution before the TCAPS architecture can be fully realized.

II. RELATED WORK

A. QoS Provisioning for Internet based Applications

The approach of moving QoS signalling from the application to the network layer has attracted some attention recently. There is an understanding that tying an application to a particular standard for QoS provision will drastically restrict its deployment options and so alternatives must be found.

The most common approach is to write the application assuming a 'best effort' service and then leave it to the user to ensure that the network capacity is sufficiently over provisioned such that QoS mechanisms are not required. This approach, although simple, is fraught with risk, particularly on consumer broadband links. Within the consumer broadband network, the Customer Premises Equipment (CPE) is the key congestion point in the upstream link, where bandwidth is at a premium and where QoS provisioning is likely to have the most beneficial impact. Consequently, alternatives to over provisioning in the 'last mile' of the network need to be considered. However, before we can do that, we need to review briefly the QoS standards that can be used in dealing with this problem.

B. Internet QoS Standards

The Integrated Services (IntServ) and Resource ReSerVation Protocol (RSVP) were defined by the IETF in 1994 with revisions to RSVP in 1997 to attempt to solve the problem of dynamic QoS provision for real-time/interactive traffic traversing the Internet. The IntServ/RSVP model uses signalling protocol messages along the network path between sender and receiver, with each node along the path storing QoS state information for each flow requesting resources.

The difficulty that has limited the deployment of IntServ is that it requires the explicit implementation of RSVP in all network nodes and possibly in the protocol stack of each end host (although the use of RSVP proxies can alleviate the latter constraint). Furthermore, each RSVP enabled network node must keep some state for each reservation request. This makes the approach unscalable for large networks with many users.

To overcome the scalability problem Differentiated Services (DiffServ) was defined in 1998 by the IETF. DiffServ enables bits that correspond to an aggregate QoS traffic class to be set in the IP header's Diffserv Code Point (DSCP) field. The DSCP is then inspected by DiffServ enabled routers along the path, which provide the QoS specified for that particular QoS traffic class.

A key problem with this approach is that every router along the network path needs to be DiffServ enabled and have the same understanding of which DSCPs correspond to which QoS traffic class to ensure proper QoS is given to the correct packet flows. Another problem is that a party trusted by the network - which is frequently not the application itself - must do the setting of the DSCP. An edge router must perform detailed packet inspection and then set the appropriate DSCP bits based on the network's QoS policy at the time. Diffserv is limited to single administrative domains unless agreement can

be reached on DSCP interpretation, and the management of edge router classification rules becomes a problem in the absence of an actual signalling protocol.

The MultiProtocol Label Switching (MPLS) Architecture [4] was defined by the IETF in 2001 to simplify the way routers make packet forwarding decisions. Instead of each router along the network path examining the IP packet header and making a forwarding decision based on this information, MPLS only requires the packet header to be analysed once. MPLS then adds a label (a short, fixed length value) to the packet, both of which are transmitted to the next hop, where the label is matched in a lookup table, a new label is added and the packet is sent to the next hop. The label is used as the basis for all forwarding decisions, and allows devices that are not capable of doing IP header inspection or not capable of doing it quickly enough to perform routing functionality. One of the benefits of MPLS is that a packet's class of service can be partially or completely inferred from the label, which allows for simplified QoS classification and management [5].

Nevertheless, MPLS still suffers from the same problems as DiffServ, in that every node along the network path has to support MPLS in order to provide QoS based on MPLS labels, and each MPLS enabled device needs to know what MPLS labels map to a particular class of service in order to provide consistent QoS. An administrator also needs to define the initial classification criteria/policies to determine which types of flows should be assigned a particular MPLS label based on their traditional IP header inspection.

An approach that avoids the application making explicit assumptions about the underlying QoS mechanism is needed.

C. QoS Enabled Products

Products have recently become available with functionality that can be used to provide some degree of QoS guarantee. Ubicom Inc.'s "StreamEngine" [6] technology, and D-Link with their "GameFuel" products [7] built upon "StreamEngine", provide routers intended for multiplayer games but which are also capable of providing benefits to other real-time/interactive traffic applications.

However, a limitation of StreamEngine is that it relies solely on local packet inspection to classify traffic into QoS classes. Packet inspection involves examining multiple fields in the packet (usually the IP header, often the transport header and optionally the application-specific transport payload) and inferring what type of traffic is being examined based on information such as "well-known" TCP/UDP port numbers, known IP addresses and data contents. There are a number of weaknesses with this technique.

The first is that it relies on being able to inspect the inner packet contents for meaningful information. Unfortunately this is not always possible (e.g. if the IP flow is encrypted).

The second problem is that well-known port numbers are not a reliable method of classifying traffic. This technique assumes that a particular port is always used by a particular application and any flows to/from the particular port involve the particular application. This assumption is often untrue.

The third difficulty is that application layer protocols can be very complex, their specifications are not always public and they regularly change. The effort required in implementing accurate classifiers and keeping them up to date is very high. Furthermore, this approach is restricted by the often very limited performance of such devices in terms of processing speed and memory.

Finally, implementations using this technique require frequent external updates to maintain the rule list, which maps packet characteristics to traffic types.

Some networking equipment manufacturers such as Cisco Systems Inc. have integrated “automated” QoS features [8], [9] into their high-end switches and routers. These features allow users to prepare one set of QoS “rules” on one device that will then distribute the same rule set to all devices on the network that are of the same type and from the same manufacturer. The system still requires user intervention to define and update the QoS policies and rules manually.

Allot Communications Ltd. has a family of products called NetEnforcer [10] on the market for IP carriers and service providers, which aim to provide simplified management of bandwidth control and service level management. The same problems inherent in the networking equipment “automated QoS” solution discussed above apply to the NetEnforcer product range. Applications requiring prioritisation on a link managed by NetEnforcer need to be provisioned manually before they are able to receive the QoS they require.

Although this work is useful and provides the necessary building blocks for providing QoS, by itself it lacks flexibility. In particular the traffic classifier needs regular updates in each CPE. An architecture in which the classification can be independent of individual CPEs is likely to be more effective.

III. THE TCAPS ARCHITECTURE

A. Motivation for TCAPS Architecture

From the previous discussion there are a number of points that have led us to develop a network-centric rather than application-centric QoS management architecture.

The first is that application developers are unlikely to include the code for explicit signalling of QoS within their application. It restricts the potential networks the application is able to run on, requires more knowledge of network protocols than the developer is likely to possess and increases the software development effort considerably.

The second factor that has led us to a network-centric architecture is that there are already a significant number of applications deployed within the network that assume a ‘best effort’ service, but could benefit from QoS provisioning. For Internet QoS to become a reality, it needs to be implemented outside the application.

In attempting to decide what capabilities the architecture should provide, we have noted that Internet links tend to be massively over provisioned within the core of the network, meaning there is minimal queuing delay and jitter introduced in the network core. The majority of network queuing delay

and jitter comes from the upstream CPE/ISP link. Consequently, providing QoS across this link is the goal of our proposed architecture. Our architecture does not change IP packet fields and does not require every device in the end-to-end path between the source and destination CPE to inspect the packets in order for QoS to be provided. Our architecture includes the automatic classification of real-time/interactive flows and enables appropriate signalling to the required devices to instantiate appropriate QoS for these flows.

Our architecture is a superset of a number of recent approaches to providing QoS. Some existing products - for example, Uvicom’s StreamEngine technology - could become an integral part of our CPE design. More importantly, our architecture will enable the central control and dynamic update of customized QoS rules for every participating CPE. The architecture we propose will still be effective regardless of whether or not real-time/interactive traffic is encrypted or running on non standard port numbers.

The architecture defines a mechanism for the automatic creation and distribution of customised rules to each CPE based on the traffic flowing to and from that CPE. No user intervention is required to define which flows contain real-time/interactive traffic, or to create, update or remove the rules that control QoS being given to these flows. (There is also potential for our system to distribute DSCP classification rules to edge routers on the ISP-side of a customer link, to further mark packets heading into the ISP’s network.)

The next section describes the capabilities needed to provide the above services.

B. TCAPS Functionality

The emphasis of the TCAPS architecture is in providing QoS controls across the ‘last mile’ between the ISP and the customer’s CPE. The TCAPS architecture comprises a centralised traffic classifier, a signalling protocol, a CPE based QoS subsystem (QoS SS) and a QoS subsystem interface.

The purpose of the traffic classifier is to identify traffic based on some characteristics. The classifier will run at a privileged point in the ISPs network, examining all packets going to and coming from each piece of CPE. Using some method of classification, traffic flows will be classified into two groups: real-time/interactive and everything else. Flows identified as real-time/interactive will be given priority over all other traffic travelling via the customer/ISP link. The minimum number of priority levels required within the system is therefore two: no priority and priority.

The QoS SS forms the basis of the traffic prioritization part of the system. The prioritisation part of the system runs at the CPE end of the link to prioritise upstream traffic. The prioritisation part of the system can also optionally run at the ISP end of the link to prioritise downstream traffic. However, this is generally not as much of an issue given the common asymmetry between upstream and downstream traffic speeds in most access networks.

The signalling protocol will be used to inform the CPE equipment (and optionally the ISP's equipment) of what traffic flows on the link are to be prioritized.

The QoS subsystem interface needs to be resident in the CPE (and optionally the ISP's equipment) to receive and implement the signalling information.

In Figure 1 we show the major flows of information necessary to implement the architecture. A traffic classifier monitors traffic to/from the CPE and identifies flows that should receive a higher priority. This information is used to generate commands to signal the CPE equipment to give those flows a higher priority. The commands are transmitted using a (new) QoS signalling protocol to the CPE. The commands are interpreted by the CPE QoS SS interface and given to the QoS SS, which then uses priority queuing or scheduling to give priority to the identified flows.

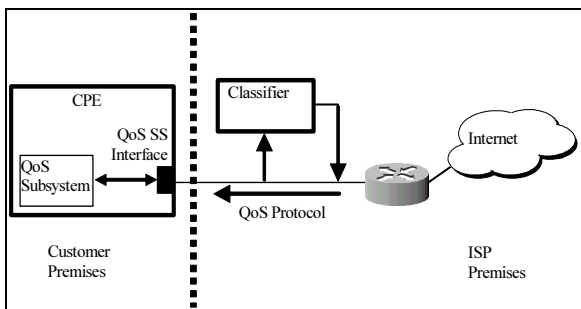


Figure 1: TCAPS Architecture – Major Information Flows

The system will function in parallel with legacy CPE to form a heterogeneous network of partially TCAPS aware devices. Devices that do not communicate using the TCAPS signalling protocol will be ignored by the traffic classifier, and will function as usual. This allows the possibility to roll out value added services incrementally without the requirement that all customers must upgrade to TCAPS enabled equipment.

Whilst providing prioritisation to real-time/interactive traffic is the current focus of this architecture, this is not the only QoS mechanism that can be used. The available QoS mechanisms are highly dependent on the implementation of the QoS SS. For example, if the QoS SS is capable of providing more than two priority levels, it is possible to establish classes of priority traffic. This can be used to express priority relationships like: VoIP before multiplayer gaming before all other traffic. Most QoS SSs also allow explicit bandwidth management, which can be used to set more traditional QoS resource guarantee conditions e.g. VoIP requires 64kbps of the total bandwidth permanently.

To allow additional features like these to be used advantageously within the system, the CPE will be able to communicate its capabilities to the traffic classifier. The traffic classifier will then be able to utilise this information to tailor rules to take advantage of some/all of the CPE's capabilities.

C. TCAPS Functional Groupings

The allocation of different functions into functional groupings will obviously depend on the size of the network. In

this section we suggest a simple grouping of functions suitable for a small ISP.

Our proposed functional grouping is shown in Figure 2. In our model we group the packet classifier into a single device - the TCAPS server. The TCAPS server receives a copy of all traffic to/from each customer through the use of port mirroring on a switch. In case the switch does not support port mirroring, other technology such as electrical or optical taps could be used to access the traffic. Larger ISPs might have multiple servers, each managing a subset of the ISP's TCAPS enabled customer base.

The TCAPS server first classifies the packets to identify whether or not the flow they belong to should be prioritized. Then the TCAPS server initiates a protocol exchange with the client's CPE to prioritise that flow. The CPE must be able to interpret the QoS signalling protocol commands from the TCAPS server. The CPE then uses priority queuing/scheduling to give a higher priority to that flow than to other flows. Regardless of how the process occurs, the application whose flows are being prioritised need not know about it. The application can be developed without any knowledge or requirements for QoS.

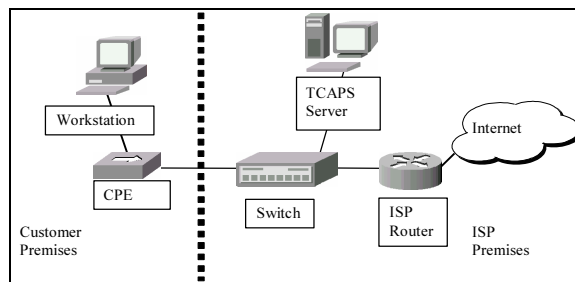


Figure 2: TCAPS Architecture – Functional Groupings

IV. PROTOTYPE DESIGN

A. Overview

In our prototype implementation of this architecture we plan to use FreeBSD [11] systems for the CPE and TCAPS server. Ultimately we intend implementing the system on a commercially available CPE, but at this experimental stage the flexibility of an easily programmed system is necessary.

CPE traffic is identified by the source/destination IP of upstream/downstream traffic respectively. An issue with this is that ISPs tend to manage IP address assignment via DHCP. It is therefore important to maintain a consistent mapping of IP address to individual CPE if accurate server to CPE signalling is to be possible. We propose that TCAPS CPE embed their MAC address in all signalling communications to the server and the server embeds the destination CPE's MAC address in all signalling communications to CPE.

For CPE to server communications, the embedded MAC address and source IP of the signalling communication can be used to build a table mapping CPE IP address to CPE MAC address. If a signalling communication is received with a

known MAC address but differing source IP, a change of IP can be inferred and the server can update its state accordingly.

For server to CPE communications, if the embedded MAC does not match the CPE's MAC, the CPE can assume to have received the communication in error and ignore it.

In the event of an IP address change, there will be a window of time where the CPE cannot receive signalling communications from the server because of the server's now inconsistent CPE IP to MAC address mapping. This length of time will be tuneable by making use of a polling signalling communication that occurs periodically from the CPE to the server. The length of time between polling communications will determine the maximum amount of time a server is out of contact with a particular CPE.

The TCAPS server can also record the state of the CPEs, enabling downloads of priority rule information following CPE or ISP outages. Our prototype will support requests for state information from the CPE and initiation of transfer of state information from the TCAPS server to the CPE.

We now describe each subsystem in more detail.

B. Broadband Access CPE

The broadband access CPE is located at the customer's end of the CPE/ISP link. This device provides the hardware and software platform for the QoS SS, TCAPS client interface and signalling protocol. Figure 3 shows a CPE block diagram.

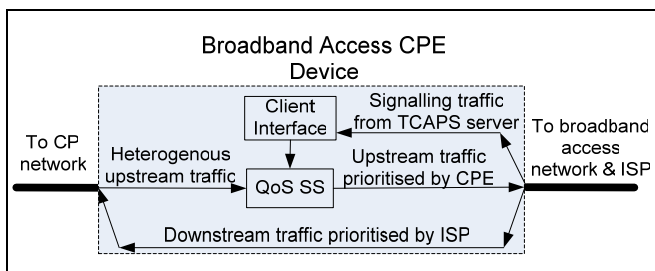


Figure 3: CPE block diagram

The CPE requires two network interfaces: one connected to the Customer Premises (CP) network, and the other connected to the broadband access network. In a standard broadband access CPE, traffic flowing from the CP network to the broadband access and ISP network would normally be queued at the CPE in a first out manner for transit onto the upstream link. This queuing arises from the fact that the CP network tends to operate at speeds above 10Mbps, whereas typical broadband access uplinks are limited to sub 1Mbps, with 128/256kbps being common among current broadband Internet access plans. Queuing at the CPE for the downstream link is unusual, as the CP network's operating speed is usually much higher than the broadband access downstream speed.

The TCAPS broadband access CPE uses priority queuing within the CPE, to ensure higher priority realtime/interactive traffic is sent on the upstream link before other traffic, even if the other traffic arrived before the high priority traffic.

C. QoS Subsystem

The QoS SS is responsible for simple packet filtering based on IP header and transport header information and providing the priority queuing framework. Figure 4 presents a block diagram of the TCAPS QoS SS.

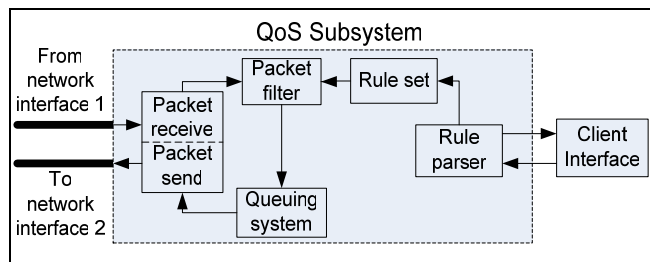


Figure 4: Simplified QoS SS block diagram

The QoS SS block diagram in Figure 4 resembles a standard packet filtering subsystem, with the addition of the queuing system and client interface blocks. A packet filtering subsystem is typically placed in the path of packet flow to restrict or modify the flow of packets between source and destination. In Figure 4, packets flowing through the QoS SS enter via network interface 1 and leave via network interface 2, after traversing the packet filtering and queuing subsystem.

The QoS subsystem is comprised of a standard packet filter as described above, with the addition of a QoS module that is capable of providing priority queuing. The TCAPS client interface interacts with the QoS SS rule parser, rather than using human intervention to manage rules. This allows remote management of a TCAPS CPE's QoS SS by a TCAPS server via the TCAPS signalling protocol and client interface.

D. TCAPS Server

The TCAPS server operates at a point in the ISP's network that is capable of inspecting all traffic going to and coming from each piece of CPE. The TCAPS server provides the hardware and software platform for the traffic classifier and signalling protocol TCAPS system blocks. Figure 5 shows a block diagram of the TCAPS server.

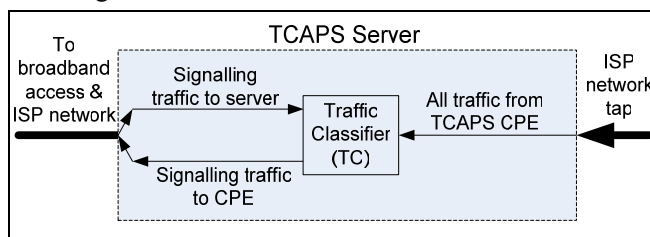


Figure 5: TCAPS server block diagram

The TCAPS server uses two separate network interfaces: one to communicate with and one to receive a copy of all traffic from CPEs. It could be argued that both requirements could be fulfilled with the use of a single network interface. However, depending on how the TCAPS server taps into the network it is impossible to use this network interface for the QoS signalling. Furthermore the use of two interfaces decreases the possibility of CPE traffic interfering with the

TCAPS server traffic (which includes the signalling between the server and the CPE devices).

The TCAPS server acts as the central management point for TCAPS related behaviour. The server is responsible for running the TCAPS system components that classify the traffic of TCAPS enabled CPE and manage these CPE via use of the TCAPS signalling protocol.

The traffic classifier system component receives traffic from CPE via a network tap from the first of the TCAPS server's two network interfaces. Its role is to classify the traffic of CPE controlled by the server. The traffic classifier also communicates with the CPE for general TCAPS administrative issues or priority rule management, by use of the TCAPS signalling protocol. All network communications are sent via the second of the TCAPS server's two network interfaces.

V. CONCLUSION

The TCAPS framework is a feasible approach to providing QoS to certain kinds of applications. It is flexible, it removes the need for individual CPE to be regularly updated with new classifier information and it removes the computation burden from the CPE. It enables CPE that are not TCAPS capable to coexist with CPEs that are, and allows the possibility of enabling QoS across the whole network. Nevertheless, there are a number of research issues that need investigation to realize the full potential of this approach.

Packet inspection is a slow and cumbersome way to identify high priority flows. It is also not necessarily a reliable one. We are experimenting with different approaches based on machine learning (see [12] and [13]). By learning the traffic patterns of certain kinds of traffic, the classifier will be able to make decisions about multiple flows much more quickly with much less information and processing. Previous work [14], [15], [16] and [17] has been done to build synthetic traffic models for real-time/interactive traffic which could be used effectively in this approach.

Although using a FreeBSD system as a CPE allows great flexibility in experimenting with different techniques, we hope that this architecture will be widely adopted. Consequently, we will investigate the use of this architecture with commercial CPE devices that support priority queuing.

Another important issue is to characterise this work for current network access technologies and extend it to those that are emerging. How effective is the approach when applied to production ADSL or cable modem based networks? What of emerging wireless network technologies such as IEEE 802.16?

Security is another issue that needs addressing within this architecture. Any external management of customer's networking equipment needs strong authentication.

Our proposed approach to providing QoS has many advantages over other attempts. The main advantage is that it removes the need for the application developer to implement any QoS functionality, and therefore also works with all existing applications. An architecture that implements this approach to provision of QoS may finally see the widespread deployment of Internet QoS.

VI. ACKNOWLEDGMENT

TCAPS originated in an undergraduate Research and Development project by Lawrence Stewart in early 2005. With the support of the Smart Internet Technologies Co-operative Research Centre (CRC), the authors are now developing an evolved version of TCAPS.

REFERENCES

- [1] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", IETF RFC 1633, June 1994.
- [2] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", IETF RFC 2205, September 1997.
- [3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for differentiated services", IETF RFC 2475, December 1998.
- [4] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", IETF RFC 3031, January 2001.
- [5] G.J. Armitage, "MPLS - The Magic Behind the Myths", IEEE Communications, vol. 38, no. 1, January 2000
- [6] Ubicom Inc., "Solving Performance Problems with Interactive Applications in a Broadband Environment using StreamEngine™ Technology", October 2004, <http://ubicom.com/pdfs/StreamEngine-WP-20041031.pdf> (as at 30/05/2005)
- [7] "D-Link 108G Gaming Router", <http://games.dlink.com/products/?pid=370> (as at 30/04/2005)
- [8] "CiscoWorks QoS Policy Manager 3.2", http://www.cisco.com/en/US/products/sw/cscowork/ps2064/products_data_sheet09186a0080091bcf.html (as at 30/05/2005)
- [9] "AutoQoS for Voice Over IP (VoIP)", http://www.cisco.com/en/US/tech/tk543/tk759/technologies_white_paper09186a00801348bc.shtml (as at 30/05/2005)
- [10] "Bandwidth control, service level management", http://www.allot.com/pages/solutions_content.asp?intGlobalId=26 (as at 30/05/2005)
- [11] "The FreeBSD Project", <http://www.freebsd.org/> (as at 30/05/2005)
- [12] T. Mitchell, "Machine Learning", McGraw-Hill Education (ISE Editions), December 1997.
- [13] S. Zander, T. Nguyen, G. Armitage, "Self-learning IP Traffic Classification based on Statistical Flow Characteristics", Passive & Active Measurement Workshop (PAM) 2005, Boston, USA, March/April 2005
- [14] T. Lang, G. Armitage, P. Branch, H-Y. Choo, "A Synthetic Traffic Model for Half Life", Australian Telecommunications Networks & Applications Conference 2003, (ATNAC 2003), Melbourne, Australia, December 2003, <http://caia.swin.edu.au/pubs/ATNAC03/lang-armitage-branch-choo-ATNAC2003.pdf> (as at 30/05/2005)
- [15] S.Zander, G.Armitage, "A Traffic Model for the XBOX Game Halo 2," (accepted for publication) 15th ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2005), Washington (USA), June 2005.
- [16] T.Lang, P.Branch, G.Armitage. "A Synthetic Traffic Model for Quake 3," ACM SIGCHI ACE2004 conference, Singapore, June 2004
- [17] T.Lang, G.Armitage. "A Ns2 model for the Xbox System Link game "HALO"," Australian Telecommunications Networks & Applications Conference 2003 (ATNAC 2003), Melbourne, Australia, December 2003, <http://caia.swin.edu.au/pubs/ATNAC03/lang-armitage-ATNAC2003.pdf> (as at 30/05/2005)