Title: Changing the internet to support real-time content supply from a large fraction of broadband residential users

John Adams
BT
Pp MLB G 7
Adastral Park
Martlesham Heath
Ipswich, IP5 3RE
Suffolk, UK
Tel: +44 1473 606 321
Fax: +44 1473 606 727

Lawrence G. Roberts
Anagran
2055 Woodside Road
Redwood City
CA 94061
USA
Tel: 1-650-906-8746
Fax: 1-650-529-9129

Avril IJsselmuiden
Institut für Experimentelle Mathematik
University of Duisberg-Essen
Ellernstraße 29
45326 Essen, Germany
Tel: +49 201 183 7667
Fax: +49 201 183 7673

Abstract

Adding QoS to the Internet raises a number of issues as to why we would want it, as well as how would we do it, and could we subtly change the user experience compared to, say, PSTN voice? Much of QoS work today is focused on improving the service over the access pipe between the end user to the ISP. But this does not adequately cover end-to-end QoS experiences for the majority of user interactions involving real-time content over the Internet. This paper explores these issues and raises the possibility of "mass content provision" as a key advantage of such end-to-end Internet QoS. It allows a large proportion of broadband users to become real-time content providers. Examples cited are: talking / explaining sequences on a home movie while pausing and rewinding, as well as selling real-time content. This paper discusses a proposed solution to adding QoS to the Internet that can be fairly easily added on to what already exists. Furthermore it is "mobility friendly". It automatically tracks & minimises flows affected by poor QoS as users move. Simulation results are presented showing the power of the method. A number of open issues and standards issues are highlighted.

1. Business Context

Today's so-called "best-effort" Quality of Service (QoS) is what real-time applications would experience when running over the Internet. This QoS is governed mainly by:

- Packet losses and delays incurred within end systems (servers, PC's, Local Area Networks (LANs), and including packetisation delays & jitter buffer delays);
- Packet losses and delays between end systems, where delays are incurred through network propagation, queuing, packetisation & at jitter buffers);
- Traffic load versus the available network capacity, influencing frequency of congestion (and packet loss) between the sender and receiver, and influencing queuing delays;
- The way that transport protocols especially TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) interact as the traffic load increases towards congestion.

There is no end-to-end co-ordination of these factors such that delay and loss target limits can be maintained. However, the resultant experience may be perfectly adequate for some types of real-time content, including voice if users are prepared to tolerate some performance degradation compared to PSTN (Public Switched Telephone Network) quality. This degradation may be perceptible only on some calls, not all calls.

We can think of examples of content that will need better than best-effort QoS (end-to-end). In compiling the list below we have concentrated on examples where the marketplace may change so that the residential user or small business becomes the content supplier.

1. Broadband voice (PSTN quality)/ video calls.

2. Video on Demand (VOD), especially if this were to evolve into a distribution model where a user gets such content from whoever has it (as opposed to getting it from a few well-known content sites).
3. The sender plays a home movie while talking over it (e.g. explaining sequences to the receiving person), pausing and rewinding to give further emphasis to the sound and pictures. In this example, a simple file transfer of the entire movie prior to speaking about it would not create the same communications experience.
4. The sender plays a commercial presentation during a conference call and each receiving person can interrupt and ask for clarification of what is going on. Again this would be difficult with reliance on file transfers prior to starting the conference presentation.
5. Improved web sites that offer good quality video images as part of their sales pitch. For example, a trial interactive sequence of a game that can be bought, or a speciality subject, e.g. "my band's music / video demo".

If the Internet was enhanced to support these types of QoS-sensitive content, then a key advantage would be the stimulation of new commercial models based on a large proportion of broadband users potentially becoming real-time content providers. We have coined the term "mass content provision" for this scenario.

The mass content provision scenario allows users to get content from a vast set of sources but creates issues on:

- how to support the QoS requirements end-to-end;
- how a reliable billing system can be achieved for this service feature.

These two sets of issues are partially connected, in that any bill directed to the service receiver should minimally depend on correct signalling procedures from the sender. This constraint, discussed further below, then shapes the choice of network mechanisms that would support the QoS requirements.

To explain this point more fully, we start with a much simpler model of content supply from within an ISP's walled garden. The walled garden is the name we will give to all the sites that can be accessed by an end user through selection of an ISP's content services. This scenario is captured in Figure 1.

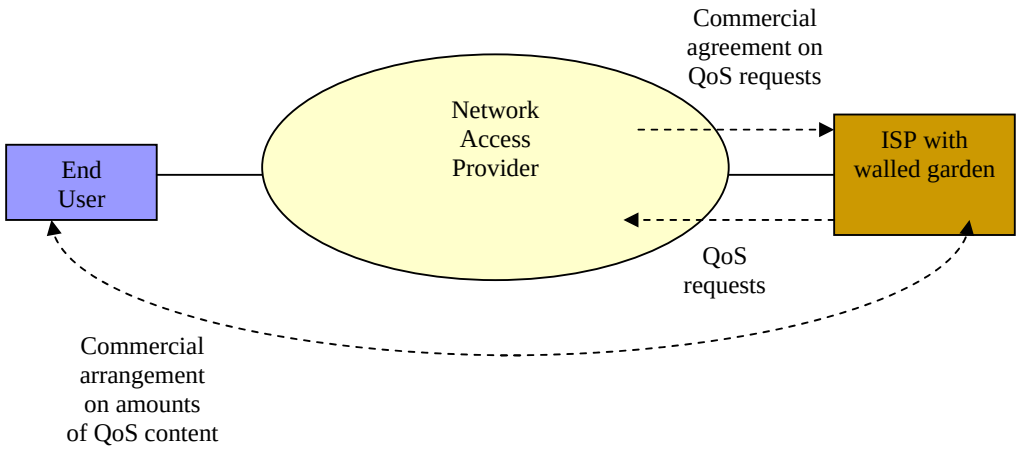

Figure 1: Walled garden scenario, showing commercial relationships

After the end user has selected some content from a specific site the ISP could:

- forward QoS requests (if required for that content) towards the network access provider;
- forward a clear-down signal after the content flow has ceased, to ensure that charging does not continue for the now unwanted network resources.

In turn, the network access provider would set up and clear down bandwidth resources based on the reception of such trusted signals. Underpinning this is a commercial arrangement between the network

access provider and the ISP. The latter is viewed as the user of a specific broadband access product that supports QoS. There may be limits or charging thresholds based on the amount of QoS content forwarded, e.g. per month. In any case the network access provider would bill the ISP in some agreed way for the network resources consumed.

To complete the description of the value chain it should be noted that there is usually no direct billing relationship between the network access provider and the end user. Instead the ISP may bill the end user by offering real-time content as an add-on service option. A value chain can therefore be established which motivates the network access provider to offer QoS-enabled products and the ISP to offer QoS-sensitive real-time content, where the service experience cannot be satisfied via best-effort.

Now lets compare this with the mass content provision scenario, shown in Figure 2. There is no walled garden and the box labelled "anyone's content" could be anywhere on the internet, not necessarily even in the same country as the receiving end user.

Clearly, as there is no walled garden, there may or may not be an ISP in the value chain between the supplier and consumer. Therefore a general model that does not assume the necessity of an ISP must assume the network responds to (and bills for) QoS request signals from either the receiver or sender.

For now, to make comparisons easier between the scenarios of Figures 1 and 2, let us assume that QoS signalling is always initiated from the source end, rather than from the receiving end user. This is the way we described things in Figure 1 – the receiving user was not the source of QoS requests. Clearly this does not take account of bandwidth boost service concepts. This is where the receiving user has a service option to temporarily increase their receiving bandwidth. However we will see from the discussion below that bandwidth boost does not adequately handle resource allocation control on an end-to-end basis capable of dealing with a general content-over-the-internet model.

A more complex issue is whether source-initiated signalling is preferred to receiver-initiated signalling. Again we will come back to this point later in the discussion, looking at the issue from the perspective of:

- how resource control can be made to easily span multiple networks;
- how service simplicity can best be achieved for the receiving end user.

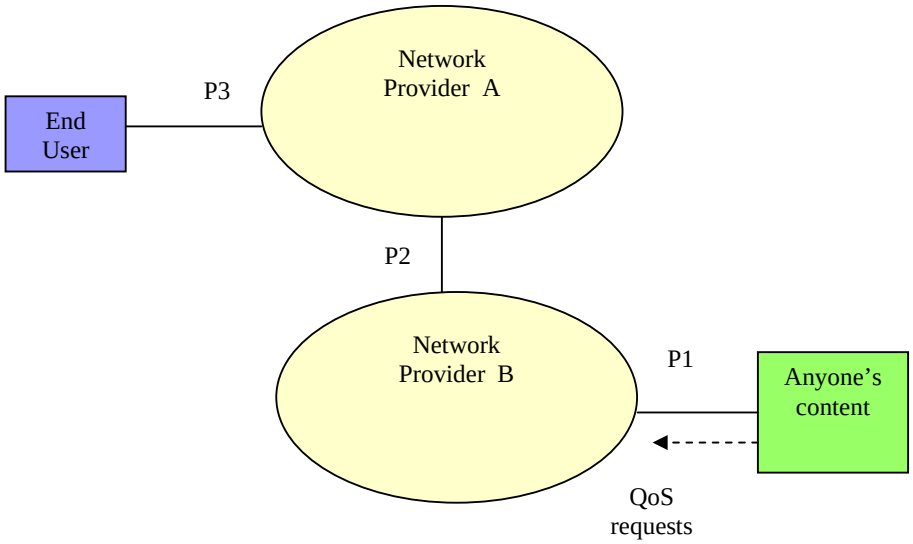

Figure 2: Mass content provision scenario

So, assuming for now that QoS signals come from the source, what are the issues with respect to reliable billing?

Two potential commercial models both illustrate the issues:

1. Real-time content supply is a billable service option offered by a basic broadband access supplier.
2. Real-time content is a billable service option of a full-services broadband ISP.

The first model has only the sender (source of content), network access provider, and receiver in the value chain and the network access provider is likely to have only a billing relationship with the receiver. In the second model there is another party in the value chain, namely the ISP and the network access provider could offer a QoS product where all its bills are directed at the ISP (i.e. as in the walled garden case).

Taking the first of these two models, and in order to offer such a service to an end user, there would need to be QoS mechanisms sited within the network of the broadband access supplier and in its partner networks. QoS mechanisms would need to be sited to control bandwidth at least at the pinch points P1, P2 and P3 shown in Figure 2. We will assume here that any bills generated reflect, among other things, the usage of scarce resources at pinch points. This creates the possibility of downward pressure on demand through pricing. Of course this may not be commercially appropriate at each and every pinch point. Nor may it be appropriate for flows of low bit rate content. Our assumption is, however, that the evolution of real-time content towards much higher bit rates creates a need for downward pressure on demand that cannot happen through a flat rate for any level of usage.

Given this assumption on the dependency between a bill and scarce network resource usage, the receiving end user needs to trust that the bills received only reflect what was wanted and consumed. This aspect needs controls on both:

- "Spam" i.e. unsolicited QoS content from a content site.
- The timely freeing-up of network bandwidth resources no longer needed by a specific end user to guarantee the low loss / delay required for real-time content.

Note that the latter point is not only important for the specific case of call duration charging, but it is clearly important that the network access provider does not have unwanted resources tied up and unavailable to meet the service expectations of other users. This is about the probability of rejection of a user's demand for service. So the timely freeing up of resources is part of the way the network access provider can keep costs down for any given target for the probability of rejection.

The second commercial model is ISP-centred, where the ISP offers QoS-enabled products supporting different volumes of real-time content. Assuming once again that the bill to the user reflects usage of scarce resources, the ISP then needs to address the same two issues above (control of spam and the timely freeing-up of resources).

We are proposing that the answer to these issues is a lightweight signalling protocol that puts minimum demands on the source content site and more controls in the access provider network or ISP network.

The proposed characteristics of the signalling protocol and QoS mechanism are described in Section 3. Prior to this we return to the comparisons of receiver-based signalling and source-based signalling.

2. End-to-end QoS across the internet

It is a significant challenge to put together an end-to-end QoS solution aimed at the migration of the Internet. The success of the Internet has in some large part been due to the relative ease of connecting networks and end-systems together to span the globe. Adding a QoS protocol that cannot easily be grafted onto what is already in place is a recipe for failure.

Having said that, there are migrations, such as the move to IPv6, that we can expect to happen and the Internet will never be a static unchanging entity. Adding QoS to the Internet will work and will be adopted if:

- adding QoS facilitates new ways of communicating that are likely to achieve popularity;
- adding QoS can be done as an "add-on" function to what is already there;
- the QoS solution minimises the complexity of settings and choices that the user must make to receive satisfactory service at an expected cost.

So imagine the scenario that, in order to get QoS, the receiving user must first select a level of bandwidth that must be reserved. Some bandwidth levels have different prices and depend on the source location. The selection process may be simplified on some content web sites but may still

require the receiving user to follow certain instructions. Next the bandwidth reservation must be requested and this must be completed in all the networks that provide the route between the content source and destination. Furthermore this must be done before the source content starts to flow, including signalling means that the source equipment understands as to when it should start to send. Finally the bandwidth reservation needs to be cleared down after the flow has finished.

Not to labour the point any further, this imagined scenario suggests that ease of use, or the lack of it, is the key to the successful uptake of QoS within the mass market. User resistance to requesting QoS may be all the stronger because of currently available real-time content offerings, such as Internet TV. Taking one such station and quoting from their publicity directed at broadband users "if there is something you want to see…just click…there is only a few seconds to wait…and the picture starts running". So why add complexity with QoS functionality? Our view is that unless real-time traffic is the subject of separate forecasting and resource management, motivated by commercial considerations, these service offerings will experience congestion. Even a few seconds of buffering in the receiver equipment will not help when there are too many simultaneous flows. We would expect this experience of congestion to be more frequent for distant sites, but even for local sites some users (who are part of a local population with above-average usage) will experience congestion at certain times of the day.

With ease of use in mind, we are proposing a radical QoS solution. We believe it is possible to deliver QoS without:

- requiring one or more network bandwidth manager functions;
- requiring either end to nominate the bandwidth required to be reserved;
- requiring either end to clear down;
- requiring either end to respond to network signals.

To achieve these aims we propose that a source generates a Start Packet ahead of the content packets. We further propose that the source is free to send content packets immediately after generating the Start Packet. The network contains new, intelligent QoS add-ons that can interpret the flow identity and security information contained within the Start Packet and handle everything else relating to the establishment and preservation of QoS. This is termed Flow State Aware bandwidth protection.

Tables 1 and 2 show some advantages of Flow State Aware bandwidth protection, over bandwidth reservation via multiple bandwidth managers (taking the general case of multiple networks between the source and destination). They also list some potential disadvantages.

Advantages	Disadvantages
Same low delay/ loss as achieved by multiple bandwidth managers, but without their complexity. Co-ordination of resource reservations between networks is unnecessary prior to sending content.	New service experience where, very occasionally, the user can get service disruption and an apology message instead of the more familiar call rejection message at the bandwidth reservation request stage.
Simplifies application usage of QoS. Just "turn on QoS" is the only requirement, with all other parameters optional.	Flow State Aware bandwidth protection will need commercial momentum to achieve standardisation and mass deployment for an anywhere to anywhere service.
Is "mobility friendly". Automatically tracks & minimises flows affected by poor QoS as users move.	
Allows pricing separation of content delivery at e.g. 1 Mbit/sec for residential users, versus the price of a clear path at 1 Mbit/sec guaranteed bandwidth.	
Rate adjustments are easy for sending and receiving users. Set-up delays should not normally be perceived.	
Very lightweight signalling to be standardised.	

Table 1: The advantages/disadvantages of Flow State Aware bandwidth protection

Advantages	Disadvantages
Hard guarantee on every accepted flow, except under network fault conditions	Call set-up delay may be perceived as significant especially for the case of viewing real-time content part way through a call.
	Changing the rate of a real-time flow may be perceived by the user as adding additional set-up delay
	Implementation complexity. Bandwidth Management is likely to be implemented with different rules and in different ways in different network domains. It also requires an out of band signalling network that is potentially more complex to standardise. However, basic standards (based on SIP) exist and work is underway to supplement and refine these.

Table 2: The advantages/disadvantages of Bandwidth Manager reservation

The detailed description of what is required within those intelligent add-ons is given in section 3 of this paper. Here we confine ourselves to an overview of the service experience that can be achieved with different levels of complexity of our total proposed "QoS toolkit". One level is described in subsection 2.1, and more advanced features are described in subsection 2.2. It will be left up to applications to decide which parts of the toolkit to exploit.

2.1 QoS denial during a flow

Referring to Figure 3, imagine a real-time content flow has been established between the source and receiving end user, passing through pinch points P1′, P2, and P3. These pinch points are associated with a direction of flow as shown by the arrows in Figure 3.

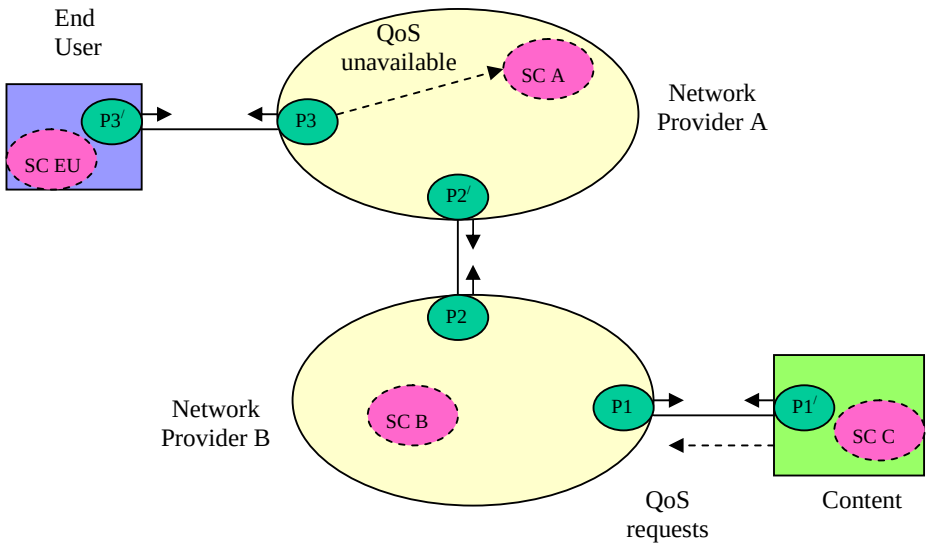

Figure 3: QoS denial during a flow

The flow had established itself by first sending a Start Packet that caused a new flow identity to be recorded at the different pinch points. Subsequent to that the flow is now sending its data packets.

Let us suppose that congestion now arises at P3. The Flow State Aware add-on function at P3 makes a local decision to deny this flow the required QoS. This means that its data packets are discarded to alleviate the congestion. P3 generates a signalling packet that flows towards the destination. This packet informs every downstream pinch-point of the service change, and the receiving user.

Optionally, a pinch-point may also inform a local service controller that a specific flow has experienced a service change. This can be used for service monitoring and billing. This message exchange (shown in Figure 3) does not need any new standards.

It is an application-level decision as to what should happen next. For example the receiver's application could send a packet to the source to cease the flow. In another instance, the preferred choice could be to send a packet to the source to request continuation with forward error protection.

Notice that we have been talking of QoS control not as flow rejection when congestion conditions are encountered (a characteristic of the PSTN network) but as selected QoS degradation. In other words a very small number of flows are selected for QoS degradation (otherwise known as "focused packet discards") so that the vast bulk of real-time flows experience no degradation when congestion conditions are encountered.

The usability of this service is significantly improved when a typical user has only infrequent experiences of such degradation. For a user's flow to be selected for degradation it must be the case that one of the pinch points has become congested. This should itself be relatively infrequent and not unlike the grade of service concept of the PSTN. In other words it is minimised through capacity planning, including forecasting. Secondly, of the possibly large number of flows currently being forwarded through that pinch point, there is a random chance of being selected.

2.2 QoS denial at the start of a flow

Referring to Figure 4, imagine a real-time content flow about to start that is passing its Start Packet through pinch points P1′, P2, and P3.

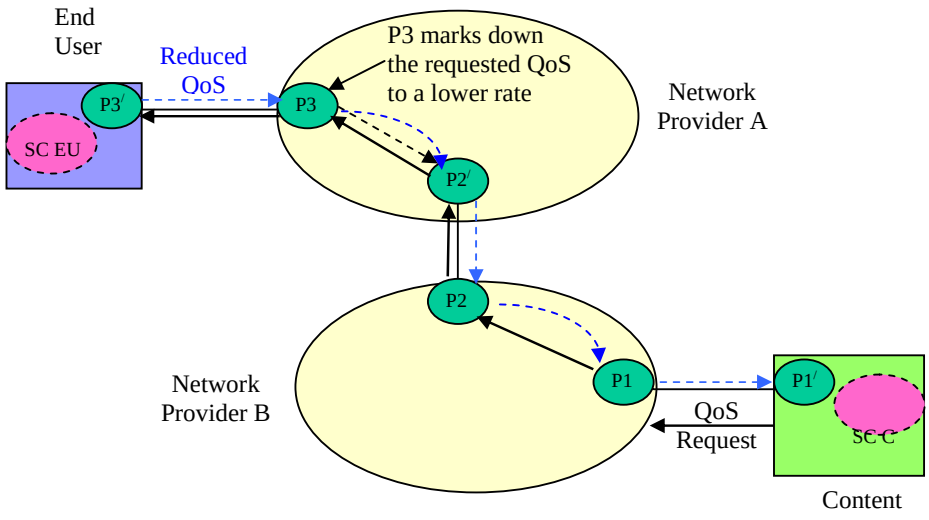

Figure 4: QoS denial only at the start of a flow

Let us suppose congestion arises at P3. In a revised version of the protocol described in section 2.1 the Flow State Aware add-on at P3 makes a local decision to deny this flow the required QoS. Consequently it marks the Start Packet with a lower QoS and continues to forward the packet, as shown in Figure 4. When the Start Packet arrives at the destination, if it is unchanged, it may be discarded. If the QoS has been reduced, it is returned to the sender. This signalling packet informs the end-point that the QoS requested is unavailable, and tells it what is actually available. Since IP usually uses different paths in each direction, the nodes in-between will most likely not see this packet. When

the sender receives this packet s/he can proceed with the reduced QoS that the path is able to support, s/he can convert to best effort, or s/he can give up.

A feature of this feedback-control scenario is that, when P3 marks down a QoS Start Packet and the receiver returns it to the sender, the sender is informed as to the QoS available over the whole path. If the sender wants to accept the connection with this lower rate or delay, s/he should respond with a Start Packet with the lower QoS. All the network nodes are now informed about the lower QoS requirement. This avoids over-commitment of the other networks such as network B. If the sender decides to abort the connection or convert to best efforts (with a different Diffserv code), s/he need not send any new Start Packet since the networks will time-out the QoS request if no packets arrive.

The reason all QoS requests need to be forwarded by the sender, is that the forward path is often different from the backward path in IP, and all the routers in the forward path need to agree and be informed. This procedure ensures that this will occur. There is a possibility that routing will change during the course of the connection due to a link failure, in which case the new downstream nodes may not know about the QoS. This is not very likely and could be fixed by the network when the connection is rerouted. However, lacking network support, if the user finds the service quality decreasing, s/he could request a new connection.

A more complex, end-to-end protocol for QoS in IPv6, TIA 1039, using a hop-by-hop option and the Flow Label in IPv6 [Ref. 1], has already been approved by the Telecommunications Industry Association. The protocol proposed in this paper attempts to accomplish much the same functionality without the full end-to-end complexity, and also to permit its use in IPv4 as well as IPv6. The major difference with this protocol, is that no acknowledgement is used and there is the possibility that the Start Packet could be lost. However, with highly reliable networks and the ability of the user to retry if something fails, the greater simplicity of this protocol and its ability to operate over both IPv4 and IPv6 has its advantages. However, due to the similarity of function and the desire for compatibility, the same message format for specifying QoS could be used here as is used in TIA 1039. This is a 16-byte message that specifies available rate, guaranteed rate, delay variance, precedence, and burst tolerance. See Reference 1 for the details of this protocol.

3. An outline of the simplified signalling protocol and QoS mechanism for the case of QoS denial during a flow.

We summarise some of the main principles:

- Bandwidth is dedicated except for a percentage of flows selected and monitored by the network. These flows may have their bandwidth removed at any time if congestion occurs at a pinch point. Typically they the last few flows that were added just prior to, and hence the cause of, congestion. The aim of this transfer capability is not to spread losses across all flows but instead to focus losses consistently on a small set of flows and provide congestion notification signals to the receiving end of such flows.
- A receiving user who does not at any time receive any congestion notification message relating to a specific flow and yet perceives packet losses on that flow above the loss limits advertised for this service would be entitled to complain.
- The percentage of flows that receive a congestion notification is a network provider's choice, although mechanisms can be established to keep it fairly small. A qualitative aim for this service is that any one end-user should not experience this bandwidth loss frequently.
- Similarly, the choice of which flow identities should be included in the monitored set is a network provider's local choice. It is likely to be random (e.g. the last few flows) but could be policy-based, allowing the possibility that flows marked with a preference marking almost never experience congestion notifications.

The network will select the flow identities of some Start Packets to be included in the percentage of flows that could be subject to bandwidth removal in the event of congestion. For all other flows, the flow identities may be held in a temporary store so that:

- New flow identities can be added if necessary to the percentage subject to bandwidth removal.
- Flows that attempt to start without sending a Start Packet can be detected.

3.1 Diffserv marking

It is proposed that networks A, B, etc each have a unique diffserv codepoint reserved for this service. It is not necessary to choose the same codepoint in each network and each network is responsible for any necessary re-marking.

If any network detects a flow of packets for which no Start Packet was seen but whose diffserv marking indicates that they require our proposed new service, then the network may choose to discard or re-mark such packets.

Packet scheduling depends on the diffserv marking of a packet. For example, packets marked as best effort class will not be subject to per-flow QoS procedures and will be scheduled such that they are delayed while real-time packets are waiting. Scheduling of packets marked as requiring our proposed new service should be appropriate for a low-delay service, for example using expedited forwarding.

3.2 Unsolicited/ unwanted flows

It is outside the scope of this paper to describe how spam flows can be intercepted and discarded. We give some possible guidelines here only as an aid to understanding all the issues that must be addressed when proposing a QoS add-on to the internet.

A possible approach is a mixture of controls that include an information element in every Start Packet supporting a security key. Appropriate key information is sent from the receiving user to be added to the Start Packet and verified at a network filter function included in the receiving user's service package. The information exchange between receiver and sender could occur while the receiver is browsing the sender's site and prior to the download. So this type of control would link quite well to content purchases.
A second control could be via an icon at the receiver which acts like a "Accept button" and informs the filter service function to permit this flow as a wanted flow. This type of control would link quite well to watching a home movie being forwarded by a relative or friend or trusted Web site. The Accept button could automatically accept requests from some sources and reject others.

3.3 Duration monitoring

The local add-on function, supervising a particular pinch point, will operate a timeout procedure to detect flows that have ceased (with local choice of how this is timeout is set but we would recommend that guidelines are established within the standards community).

When a flow is deemed to have ceased then any further packets are treated as a flow without a Start Packet and the packets may be discarded / re-marked. However we recommend, and discuss further in section 3.5, that all such cases are dealt with by the network inserting a new Start Packet into the flow whenever it detects such a condition. This is so as to deal with silence-suppressed voice, mobility, and other applications that may innocently cause this condition to happen.

3.4 Congestion Notification Packet

When a network controlling some pinch point is required to focus packet discards on a flow then it generates a Congestion Notification Packet. This is forwarded to the receiving end. The Congestion Notification Packet should be standardised to allow downstream pinch points to read it and for it to be recognised by the receiving end system as a standard network signal.

A downstream pinch point may choose to respond to a Congestion Notification packet by placing the associated flow identity in its focused discard register (also called the Drop Window – see section 3.5).

How the receiver application responds to a Congestion Notification Packet may differ depending on the application. It could forward a message to the source to cease the flow. The receiving client could also display an apology message to the user. This is outside the scope of this paper.

3.5 QoS operations at a local pinch point supervisor

A full description of these operations is given in [Ref: 2]. The following gives an overview of our proposed per-flow QoS-control principles. These controls are only applied to packets that have been given a diffserv marking indicating this new service category.

First, to explain how focused discards underpins the provision of guaranteed bandwidth, initially suppose (for simplicity) that all content requiring this service is constant bit-rate. Assume also that new flows are added one at a time and with some short interval of a few seconds or more before the next flow is added. Then, as the load increases, congestion happens through the "last straw" principle. In other words, there is no congestion until a flow is added which takes the combined bit-rate above the capacity available. Furthermore, by deleting the packets of this latest flow congestion would be removed.

This suggests that if we had a device (which we will term a "Pinch Point Supervisor") that could:

- memorise the identity of the latest flow (at least);
- be prepared to delete the packets of the latest flow (at least)

it would then be possible to protect the remaining flows from the effects of congestion.

We will call the memory area the "Flow Register" as shown in Figure 5. Most of the additional complexity comes from the fact that we do not assume a simple constant bit-rate world for real-time content. The presence of variable bit-rates causes some uncertainty, because a flow can be added which does not immediately transmit at its peak rate. The next flow could then be added still without any apparent signs of congestion. Now suppose the earlier variable-rate flow hits its peak rate. Simply deleting the last flow is not guaranteed to recover from congestion.

This suggests that a better principle may be to memorise the last "r" flows. We should try to delete all packets of the latest flow first (as this is still the most likely cause of congestion) but be prepared to delete the packets of the other r-1 flows if congestion persists.

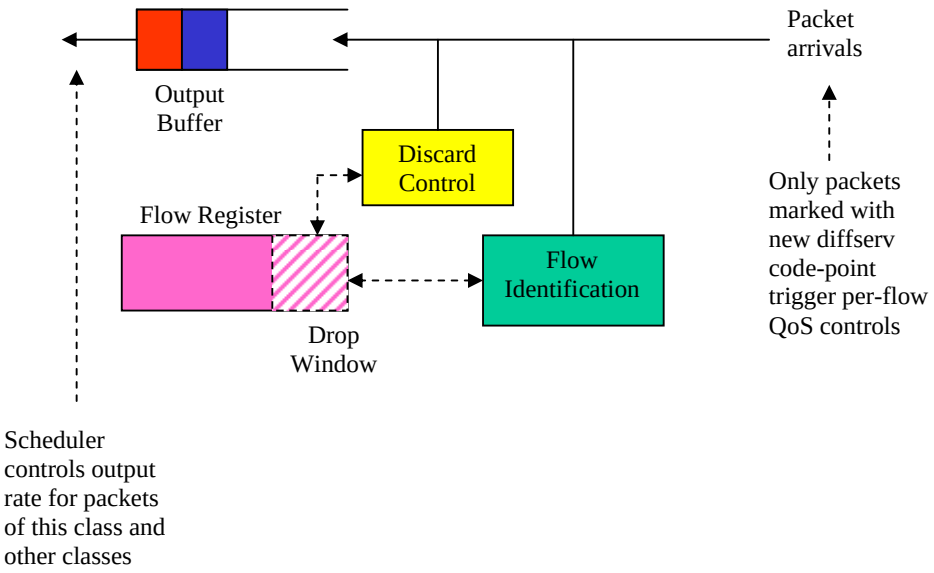

Figure 5: Local QoS control at a "Pinch Point Supervisor"

We term the set of r flows that are subject to potential focused discards the "Drop Window" (shown in Figure 5). We may think of flows in the Drop Window as those that the network has not yet accepted as guaranteed. There is no requirement for the network to signal this status to the receiving user as all flows generally start in this status (the exception being flows marked as preference flows through information contained in the Start Packet).

The in-band Start Packet contains all the information necessary to identify the packets of any flow, e.g. source and destination addresses, flow label or port numbers. It also provides for a preference flow indication. When the pinch point supervisor (PPS) receives a Start Packet for a new flow, it stores the flow identity in the Flow Register, usually in the Drop Window.

The Drop Window has only a small number of entries. All new flows must spend some time in the Drop Window until they have each sent "x" packets (a parameter set by the Administrator). With the duration of time, flows gradually move through the Drop Window towards guaranteed status. This "movement" within the Drop Window actually consists of a re-classification of flows so that they become less likely to be targeted. The latest flow will be the subject of immediate discards in the event of congestion and only when discards on this flow fail to reduce congestion will the additional flows be targeted.

Guaranteed status implies removal from the Drop Window. Such flows will not be the focus of packet discard any longer, except in cases of very rare congestion. A rare condition, termed "emergency discard", can occur where the amount of packets necessary to drop (to recover from congestion) exceeds the amount of packets available within the Drop Window. For example:

- a fault condition causing re-routing and partial reduction of the available bandwidth;
- a bursty source with a large peak rate that has guaranteed status but has been silent for some time. When it becomes active again its additional packet load is greater than the amount of packets available for deletion within the Drop Window.

These cases should be very rare. In fact the initial size of the Drop Window should make this very rare, and if all flows were CBR, it would occur only through fault conditions causing partial loss of bandwidth. In such cases, the "emergency discard" process randomly selects one additional flow and adds it temporarily to the Drop Window, thereby increasing the packets available for discard. This procedure could be repeated if necessary until congestion ceases. It is not anticipated that this procedure would cause a noticeable difference to any user's frequency of experiencing QoS denials.

The pinch point supervisor (PPS) detects congestion by monitoring a token bucket fill level (included within the "Discard Control" area of Figure 5). Added tokens represent the current arrival rate. Tokens depart at a rate that may be set slightly lower than the true rate available to the scheduler. If the fill level of this bucket reaches a certain configured threshold, it triggers packet discard. This is applied to all packets of the latest flow added to the Drop Window. Furthermore, the receiver of the flow, and the network billing point, are notified about the congestion by a Congestion Notification message.

If congestion continues and the token bucket fill level reaches a second threshold, then the PPS begins discarding packets of the other flows in the Drop Window. Again Congestion Notification messages are sent to the receiver and network billing point.

If congestion causes the token bucket to reach an upper threshold at an even higher fill-level this triggers the "emergency discard" process and the selection of another flow from the guaranteed set to be added to the Drop Window, as already discussed above.

Flow identities that are not in the Drop Window could remain in the Flow Register for policing purposes if required. For example if there is a concern that flows have started, perhaps in error, without first sending a start packet (for IPv4). They may also do this deliberately to avoid being placed in the Drop Window and, if enough users did this, the service would gravitate towards the "emergency discard" process of selecting random flows to be dropped. It may be acceptable to run the service without checking for Start Packet violations and accept "emergency discard" random flow selection for focused discards. But, if the preferred principle is that those flows that have been active for longest should be the least likely to experience discards, then Start Packet violations need to be policed. This type of policing does not need to be invoked at every pinch point. Typically it would be implemented at access interfaces and network-network interfaces in each Service Provider's network.

At such a policing point, each packet flow identity is compared with all the flow identities in the Flow Register. If it does not match one of these identities it is deemed to be in violation. A violating flow could be subject to immediate discard or re-marking. However we recommend a “mobility friendly” mode of policing, such that the policing point inserts a Start Packet into violating flows. This creates the correct Drop Window tracking of moving flows, and minimisation of flows affected by poor QoS,

Flow identities are eventually removed from the Flow Register via a timeout procedure that detects when a flow can be assumed to have ceased. This is also important for mobility, as it clears out Flow Registers of flows that are no longer routed through that PPS.

If policing is limited to the detection of a flow that starts without a Start Packet then have we done enough to limit the effects of all unexpected user behaviour? For example, what about a case where the source sends a Start Packet but transmits the data packets at a rate much bigger than advertised in the Start Packet? We recommend the following additions to cover this case:

- The Start Packet rate availability check (see section 2.2) be treated by the network as a query that assists an application but does not mandate any rate limit;
- In any case, where the *network* inserts a Start Packet, having detected a flow that apparently has started without one, the advertised rate should be marked as the default rate. There could be no meaning attached to the application violating this default rate. Note here that the effect of a network-generated Start Packet may trigger the source application to generate another Start Packet, e.g. to re-advertise the preference marking;
- That focused discards are the QoS mechanism that deals with any congestion conditions arising from the dynamics of rate and/ or mobility-based changes in a flow, but that rate increases are limited as follows:
- The network provider specifies two aggregate rate limits (statically assigned maximum permissible rates) at the user-network interface (UNI) that apply to both the total sending rate generated by all flows of this service and the total preference packet (high priority) sending rate. Policing of these aggregate rates would apply.

Before we finish with the subject of unexpected behaviour, users may see an incentive to maintain existing flows long after they are needed through applications that send dummy data flows continuously although to the network they look like real data flows. This is effectively another way of avoiding the Drop Window except for the first relatively short period after the flow is established. Possibly the most effective mechanism to combat this behaviour is an incentive to finish a flow via a billing solution.

4. Simulation Results

In this section, we discuss the results of our simulation model. The heart of this model is the simulated functionality shown in Figure 5. It was originally created at the Institut für Experimentelle Mathematik, University of Duisberg-Essen, to check the logical operation of the proposed function as it transitions between states of “normal”, “discard”, and “emergency discard”. However its use has been extended to look at the influence of particular threshold trigger values. We also wanted to look at some tests of the overall logic, starting with some fairly pathological cases. Some of the results of this current phase of simulation studies are presented here.

A model was implemented, with a number of traffic generators attached, capable of transmitting CBR or VBR flows at different rates. The model allows us to look at packet losses on each of these flows during moments when a 10 Mbit/sec link is driven into congestion. Of course this is a very low link speed but it allows simulation run-times to be shortened and is a worthwhile study point in its own right, relating to some broadband access conditions. In the first set of simulations, we discuss the results of CBR-only traffic flows and, in the second section we discuss the results of VBR traffic flows.

4.1 CBR-only simulations

Congestion was simulated by using 10 x 1 Mbit/sec CBR generators, started at random times, plus 1 64kbit/sec CBR generator, started last so that it is the reference stream. Packets had a constant length of 680 bytes for the 1 Mbit/sec generators and 120 bytes for the 64 kbit/sec generator.

The pattern of packet arrivals is shown in
Figure 6. This shows that in every .005 second time-slice, the simulated traffic arrives in bursts rather than completely smoothly. This occurs because some flows are generating their packets near-simultaneously leading to clusters of packet arrivals at the congested link.

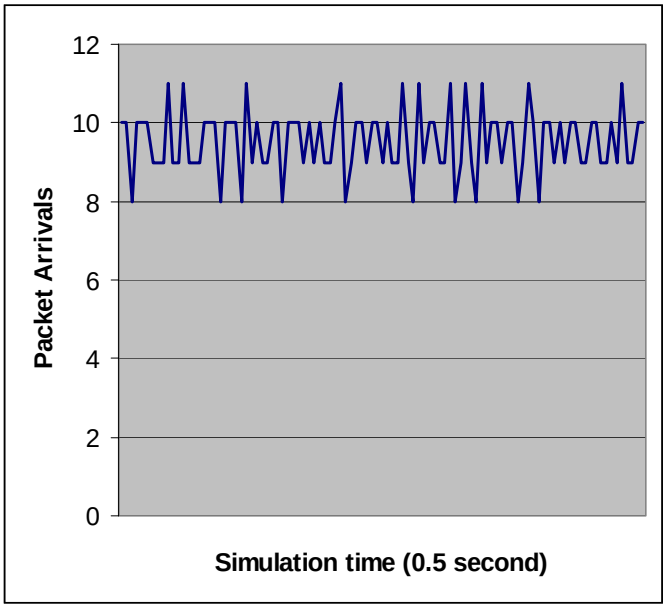

Figure 6: Packet arrivals for CBR generators, every .005 s, over a half-second period

Clearly we want the system to behave such that when the 10 Mbit/sec link is loaded only by 10 x 1 Mbit/sec flows there are no packet discards. Because packets arrive in bursts it is necessary to set the buffer so that the lower threshold (triggering the onset of discards) allows the buffer to absorb such bursts. This is a normal design consideration for any packet network supporting loss-sensitive flows. For the particular scenario that we simulated when the load was 10 x 1 Mbit/sec and the lower threshold was set at 3 x 680 bytes or above, there was no packet loss. When set at 2 x 680 bytes, there was packet loss.

Of course the load we have simulated is in no way representative of real traffic in the network except possibly on a broadband access link carrying several CBR flows. However the principle of designing a loss-free service that tolerates a defined level of packet bursts can be demonstrated even with our choices of generators and burst effects.

A 64 kbit flow was then added to give congestion. This was the last flow to arrive on a link that was successfully supporting packet loss-free transmission of the 10 x 1 Mbit/sec flows. The policy or preference setting we assumed for this simulation was that this latest flow should be subject to discards if congestion is detected. The flow identity of the 64 kbit/sec stream was added to the Drop Window. We were looking for one of two alternative results to be demonstrated:

1. that, if we only allowed one flow identity to be retained in the Drop Window and if the latest flow is a relatively low bit rate, does the discard control function cope correctly in terms of focused discards on just one flow?
2. Alternatively would we see the discard control function invoking emergency discards which is its graceful process to recover from sustained congestion. Emergency discards would be applied first to a randomly-chosen flow, so that 2 flows experience discards.

Our results showed that, over a 30 minute simulation period, the only flow to lose packets was the 64 kbit reference stream. No flows were added as emergency delete flows. Changing the seed still gave the same result. Figure 7 shows the ratio of protected flows to disrupted flows, when using the

protocol. The reference flow transmits approximately 66 packets per second, (119,999 packets in a 30 minute period), and lost 119,121 packets from its flow – 99.98 % of the packets were therefore deleted.

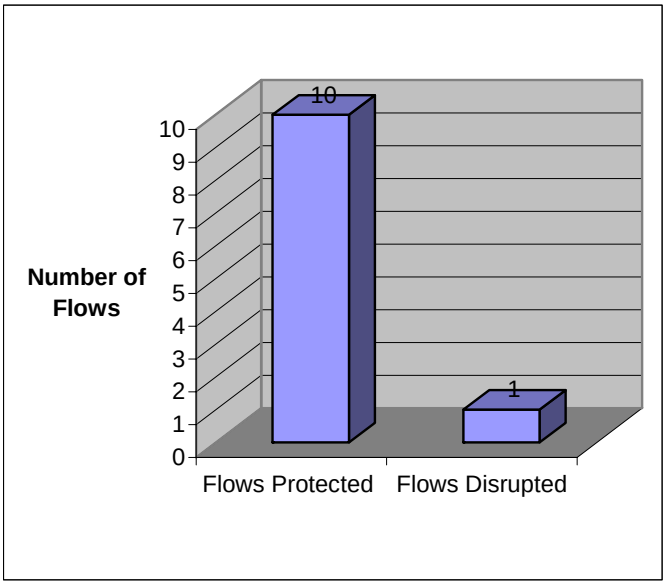

Figure 7: Ratio of protected flows to disrupted flows when using the protocol

Figure 8 gives the average queuing delay through the buffer for CBR traffic at the moment when the 10 Mbit/sec link is congested. Obviously lower delays would be experienced when the link is not congested. Also these delays would be very much lower on a high-speed link.

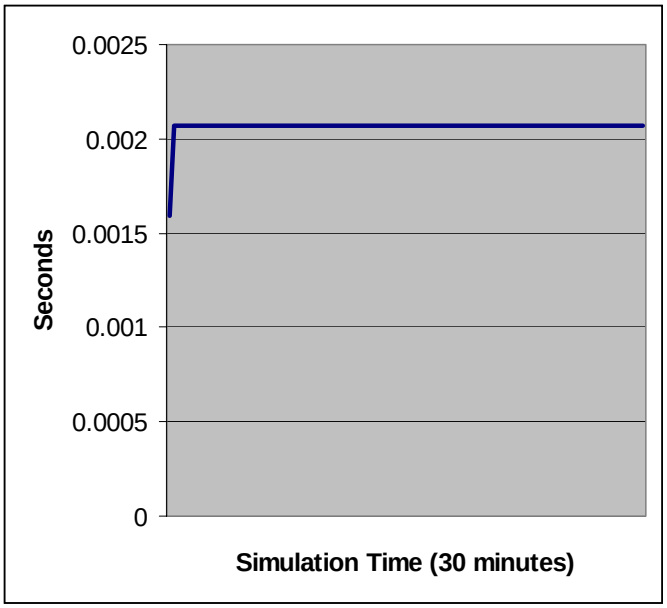

Figure 8: Average queuing delay through buffer

4.2 VBR simulations

With VBR simulation we were looking for how the packet deletion process copes when a VBR flow is currently "quiet" (no packets arrive) at a time when several new flows are attempting to start up. Again we used a 10 Mbit/sec link rate.

Generator	Type	Rate	Pkt Size (bytes)
1	CBR	1 Mbit	680
2	CBR	1 Mbit	680
3	CBR	1 Mbit	680
4	CBR	1 Mbit	680
5	CBR	1 Mbit	680
6	VBR	3 Mbit *On: mean 1* *Off: mean 0.5*	680
7	CBR	1 Mbit	680
8	CBR	1 Mbit	680
9	CBR	64 Kbit	120
10	CBR	64 Kbit	120
11	CBR	64 Kbit	120

Table 3: The traffic generators

The generators were started in the order shown in Table 3, at varying times. The last 5 flows (generators 7-11) were deliberately set to start up in a silent period of the bursty generator. Clearly, during such a silent period, the total load on the link is 7 x 1 Mbit/sec + 3 x 64 kbit/sec which would not trigger any packet losses. However, when the VBR flow returns to activity, the link will be congested by the extra 3 Mbit/sec of load.

We wanted to examine two main effects.

1. To deliberately trigger the emergency discard behaviour of the packet deletion process by having only one or two flows in the Drop Window. Here, we were looking to see that the emergency discard process is "graceful" in that it tries to keep the total number of flows experiencing discard to a minimum (albeit using a random choice of which ones these will be). We did not put any preference control into this choice such that certain preferred flows are not eligible to be chosen (for example in a preference setting required for military purposes).
2. To look at how discards varied with different choices of the lower (discard starts) threshold and the higher (emergency discard starts) threshold. Also to look at whether the emergency discard process managed to stop any packet being lost through buffer overflow. It should be noted here that we did not use a token bucket with a lower leak rate than the link rate, so buffer overflow was not protected by any such setting.

A snapshot was taken of the pattern of packet arrivals, for .005 second time-slices, over a 1.5 second period. This is shown in Figure 9. This shows that packets arrived in clusters of up to 11 packets in each time-slice. Clearly visible is the variation in arrival rate due to the VBR flow, part of the time being in the "off" state, and the rest of the time being in the "on" state. During an "on" state, packets arrived in clusters of 8 – 11 per 0.005 second time-slice. We would expect an average of around 10 packets per 0.005 seconds during such an "on" state, given that most packets are 680 bytes in length.

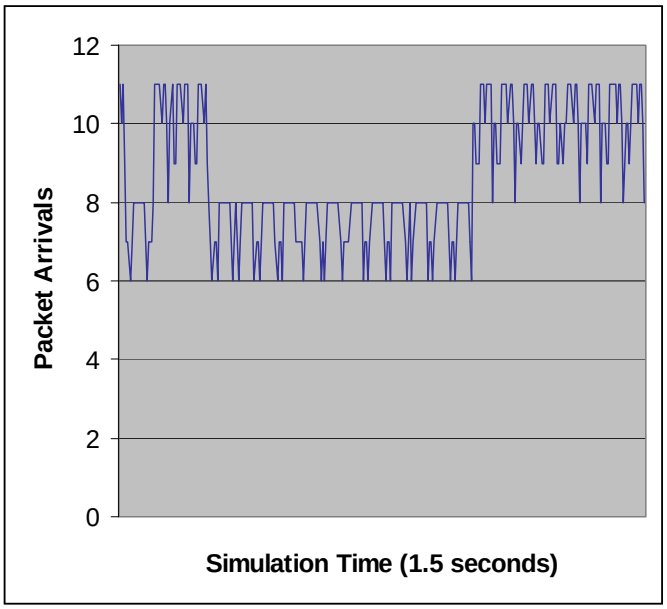

Figure 9: Packet Arrivals for VBR traffic in .005 time slices, over 10 seconds

Buffer	11 x 680 bytes	59840 bits
Lower Threshold	6 x 680 bytes	32640 bits
Upper Threshold	9 x 680 bytes	48960 bits

Table 4: Experiment 1 parameters

Buffer	11 x 680 bytes	59840 bits
Lower Threshold	5 x 680 bytes	27200 bits
Upper Threshold	8 x 680 bytes	43520 bits

Table 5: Experiment 2 parameters

As shown in Table 4 and Table 5, the first experiment had slightly less aggressive settings than the second experiment. One reference flow (64 kbits) was kept in the register window. If discards on this flow was insufficient to control congestion, an emergency flow would be added. Again, if that still proved insufficient, a second emergency flow would be added, and so on. Given the level of overload in this experiment, we would expect discards to be insufficient without emergency flows. For "experiment 1" threshold settings, our simulation runs resulted in only 1 emergency flow being added, and this was sufficient to control congestion. Flow selection for this emergency flow is based on the first arriving packet after the upper threshold is triggered. So it is not random across all flows but is biased towards higher bit rate flows. We wanted to see how badly the flows were disrupted by discards (could they conceivably have continued offering a somewhat degraded but acceptable experience to the end user?). So we also looked at the total packet losses in the emergency and reference flows.

For "experiment 2" threshold settings, sometimes 1 flow was added, and sometimes 2 flows were added. Although we then changed the Drop Window so that 2 flows were retained, the model still picked up an emergency flow, so that in total 3 flows were disrupted altogether.

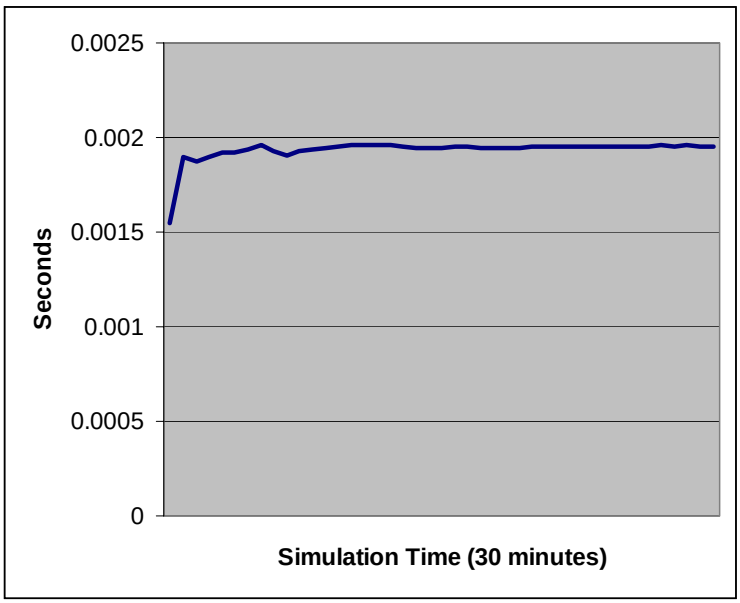

Figure 9: Average queueing delay for "experiment 1" thresholds

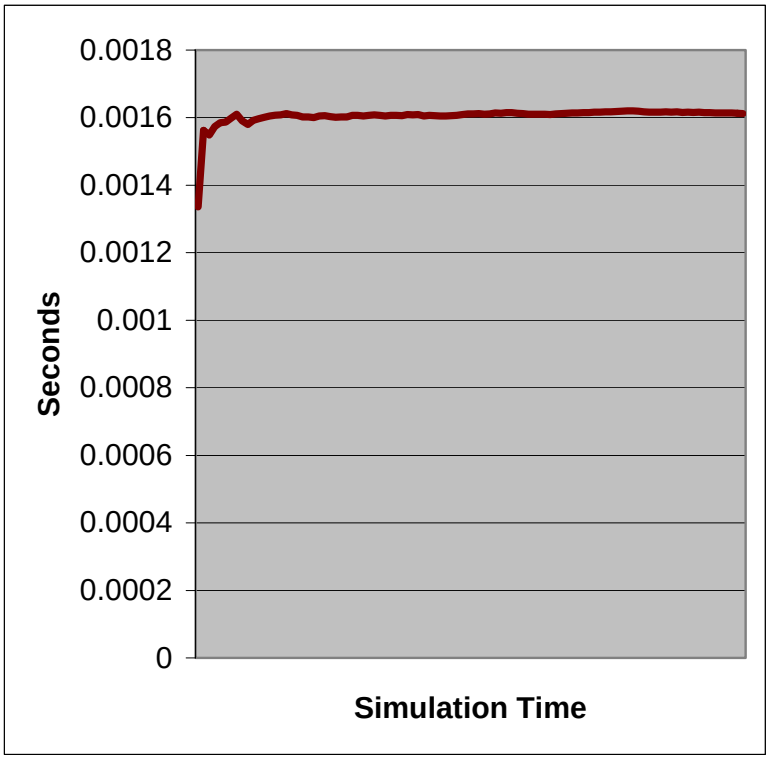

Figure 10: : Average queueing delay for "experiment 2" thresholds

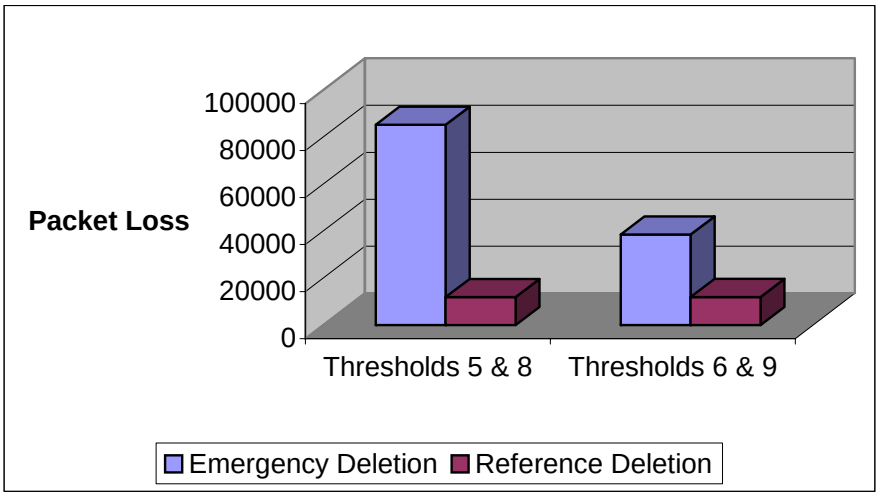

Figure 11: Packet Loss

Figure 11 shows the packet loss for experiments 1 and 2. This may also be expressed in terms of a 95% confidence interval to give packet loss values as shown in Table 6:

Experiment	Reference Flow	+ / -	Emergency Flow	+ / -
1	50392.8	2113.9	38458.6	1109.6
2	11911.4	2846.6	85143.2	7833.1

Table 6: Packet losses for the reference & emergency discard flows, with confidence intervals

Clearly experiment 1 settings reduce the total disruption to user experience. For both experiments 1 and 2, no packet losses occurred due to buffer overflow.

Figure 12 shows the ratio of protected flows –v- disrupted flows for experiment 1, and Figure 13 shows the ratio of protected flows –v- disrupted flows for experiment 2.

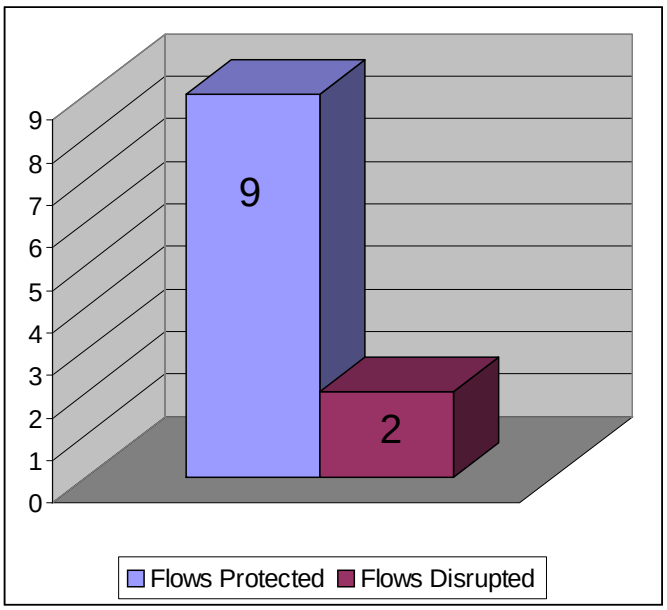

Figure 12: Ratio of protected flows to disrupted flows for experiment 1

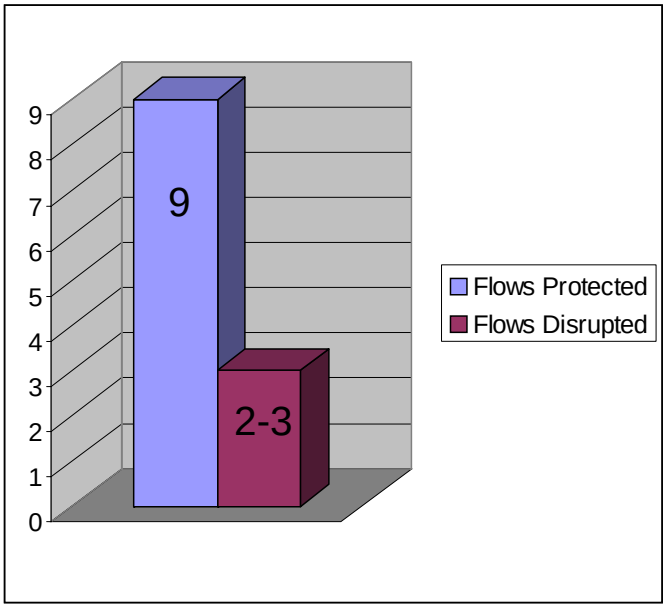

Figure 13: Ratio of protected flows to disrupted flows for experiment 2

From all these results can be deduced several things:

- First of all we are aware that currently we are only able to report on the simulation results of a few pathological cases and that more work should be done on simulating real traffic scenarios. But the results we have are encouraging, especially given that have focused only on the simpler part of our total "toolkit" (the part described in subsection 2.1.)
- The discard process gracefully backed off (i.e. it always kept the number of flows affected at near minimum) when unforeseen overloads occur, at least in our VBR simulation scenarios. The degree of preservation of flows and the number of packets discarded is dependent on the threshold settings but this may have a less important effect where the differences are small in terms of the total number of flows disrupted.
- We have not yet tried a link failure scenario where traffic is re-routed to another link causing perhaps massive congestion. The main focus of our studies have been on recovery from conditions that push links slightly over their load limit and into congestion, as could be anticipated from most normal CBR/ VBR mixed traffic scenarios under non-fault conditions. The main issue (perhaps) with major bandwidth losses is the cycle time or "thinking" time of the emergency discard process as it attempts to decide whether to add yet more flows to its Drop Window because congestion has not yet disappeared. We believe there are many ways that this condition could be recognised and optimised. For example, by making emergency discard pick first one flow, then two, then four, etc or an even more aggressive ramp. However, this remains an item for further study.
- The results also demonstrate what happens if the source changes its rate to a higher rate, giving rise to congestion at a pinch point. It is similar in nature to the effects of our on-off VBR flow. The network gracefully recovers with only a few flows being subject to discard but there is no fairness principle involved such that the new higher rate of the flow is bound to make it the subject of discard. All flows are treated as equally vulnerable, except for preference marked flows.

5. Conclusions

As mentioned in the preamble to sections 2.1 and 2.2, we have introduced a new "toolkit" of network QoS control functions and associated service request signals, allowing applications to choose which features they need. We have demonstrated that it is possible to support real-time content delivery without:

- requiring one or more network bandwidth manager functions;
- mandating either end to nominate the bandwidth required to be reserved;
- requiring either end to clear down.

Furthermore it is left up to applications to decide whether to exploit richer mechanisms of the toolkit (see subsection 2.2) or not and, if not, then QoS support is still possible without:

- requiring either end to respond to network signals.

We propose these mechanisms as a candidate toolkit for QoS support across the Internet. Furthermore we believe this toolkit is "mobility friendly" in the sense that it automatically tracks & minimises flows affected by poor QoS as users move.

We believe these QoS mechanisms could be deployed in a small way initially. For example users could be offered a "broadband plus" service in certain early adopter locations, where "broadband plus" allows them to receive or send packets of this new service type. If such users look for source sites supporting this new service then QoS-enabled flows could pass between them.

Initially the QoS mechanisms may not be deployed everywhere and possibly only protect the access links of the two end-points. In its early evolutionary phase the service may be best suited for exchanges of QoS-sensitive content within local or regional communities, between friends, family members, etc.

Another early contender for deployment of this functionality could be wireless hot-spots, where it could allow multiple users to experience real-time flows that don't degrade.

Eventually, users may migrate to ISPs that offer extended coverage via Internet peering centres supporting this capability. The end result should be that unlike today, users can be sure that when they start watching a video over the Internet or engage in a video conference, it will be very unlikely that the QoS will degrade half way through.

The underlying assumptions are that:

- If either or both of the two end-points are on broadband XDSL access links, then our proposed functionality should be deployed at the DSLAM pinch point for upstream flows (i.e. flows going towards the network). Initially this functionality may only be deployed at early adopter sites.
- The technology of per-flow processing should scale and give rise to a simpler-to-use real-time QoS on the Internet. The simulation results show that QoS guarantees can be maintained with very few flows in the Drop Window. This aspect of the proposal appears to scale very well with any size of network and total number of flows. Policing of Start Packets at key interfaces places the biggest demands on scale, noting that we have not identified a requirement to police rates on a per-flow basis.

Capacity can be planned so that the typical user experience sees very infrequent denials of QoS. The value of the discard mechanism is to handle not only the expected but infrequent congestion arising from the planning rules. It also handles the unexpected and can look after another aspect in the value chain, namely the maintenance of preference flows during such congestion events.

There are standards issues that must be addressed. These include the Start Packet and Congestion Notification Packet. Also included are the preferred settings of parameters like the flow cessation timeout. Agreement also needs the active involvement of the end systems vendors who will develop the applications and usability of the service.

Overall we believe this functionality will enrich the communication experience and content provider aspirations of a large fraction of broadband end users.

References

1. IETF Draft of IPv6 QoS Signalling: http://www.packet.cc/IPv6Q-IETF-2A.htm (Similar to TIA 1039) by Lawrence G. Roberts, Nov 1, 2004
2. J.L.Adams & A.J. Smith**.** European Patent Application No EP 01 30 5209. *Packet discard control for broadband services* Issue June 2001