An Experimental Study of Assured Services in a Diffserv IP QoS Network*

N. Seddigh, B. Nandy, P. Pieda, J. Hadi Salim, A. Chapman  
Computing Technology Lab, Nortel Networks  
Ottawa, Canada  
Email: {nseddigh, bnandy, ppieda, hadi, achapman}@nortel.ca

Abstract

Much attention has recently been given to the Differentiated Services (Diffserv) approach to provide Quality of Service for IP networks. This packet-marking based approach to IP QoS is attractive due to its simplicity and ability to scale. Two of the most popular services proposed for the Diffserv approach are the Assured and Premium Services. In this work, prototypical implementations of Diffserv components are described. The prototypes are used to study the single-queue, dual drop-preference model proposed as a basis for Assured Services in Diffserv.

RED (Random Early Detection) is used as the packet discard algorithm in Assured Services. Three different models of RED are experimented with to observe their implications on providing Assured Service. Secondly, experiments are performed to determine service discrimination that can be achieved for Assured Service traffic using this approach. Finally, experimentation was performed to observe the behaviour of real-time low-bandwidth (eg voice) flows that are provided the benefit of an Assured Service.

KEYWORDS: IP QoS, Diffserv, Assured Services, Edge, Core, RED, Voice, TCP, UDP

1.0 Introduction

Traditional IP Networks offer users best-effort service. In this model, all user packets compete equally for network resources. The best-effort model has been sufficient until recently when the usage and popularity of IP networks has soared. This rise in usage and popularity has been paralleled by a rise in user expectation regarding the type and quality of applications that can be run on these networks. Unfortunately, due to limited networking resources (bandwidth, buffer constraints), sporadic and periodically sustained congestion is an eminent feature of current IP networks. In 1995, a large-scale study on traffic characteristics at 35 Internet sites showed that on average, 5% of all transmitted packets were lost and that majority of the loss could be attributed to congestion at bottleneck points [21]. In “busy” connections, the loss rate jumped to over 9%.

Congestion of this nature does not encourage mass adoption of IP networks as a transport mechanism for real-time and business deemed mission-critical applications. Developing improved and reliable service predictability for IP networks is one of the key challenges as service providers attempt to lure business and leisure applications on to the Internet. The underlying concept in IP Quality of Service (QoS) is the ability of network operators to offer differing levels of treatment to user packets based on their requirements.

Initial attempts to provide QoS in IP networks can be traced back to RFC791 [11]. This ‘Request For Comments’ (RFC) defines the TOS field in the IP Header and associates TOS markings with various alternative services offering lower delay, loss or cost. It was intended that Routers would use these TOS markings to provide different treatment to packets. A more thorough QoS architecture was defined in the Internet Engineering Task Force (IETF) Integrated Services Architecture (IntServ or ISA) working group, where work began in 1994.

* Submitted to SPIE symposium on Voice, Video and Data Communications: (QoS Issues Related to the Internet), November 1998
IntServ [5] was the IETF’s first substantive effort towards providing QoS for IP networks. In IntServ, end-host applications explicitly signal their QoS requirements (using a protocol such as RSVP [6]). Each Router along the route would review this request and an end-to-end path for that transaction (flow) pinned down. The Routers along that pinned-down path check the RSVP QoS request against a policy module and available resources. If resources were available and the policy module granted the request, the resources were reserved for the duration of that transaction.

The key benefit of the IntServ approach is that it can provide hard QoS guarantees as required by ‘quantitative’ applications such as voice or video. Though quite mature in terms of its specification, to date, IntServ has not been deployed on a wide scale due to a number of drawbacks:

- **Scalability** - IntServ requires intermediate Routers to maintain states on every single flow passing through them. Such a requirement is not realistic for core Routers in the backbone network due to the sheer volume of flows.
- **Overhead** - IntServ does not provide a solution for what presently constitutes a large proportion of Internet traffic - short flows with elastic behaviour. The extra task of end-to-end resource reservation is too large an overhead for flows that typically transmit no more than a few packets.
- **Route Pinning** - IntServ requires nailing down of a path between end hosts. This connection-oriented approach is foreign to IP Networks.
- **Host Orientation** - IntServ also focussed very much on host-to-host connections and did not allow for the fact that many service agreements would be at a more aggregated level with intermediate management points such as the gateway between an enterprise and the IP network.

Based on the IntServ experience, work started on a new approach to IP QoS. The Differentiated Services approach proposes a more scalable way to deliver IP QoS. It operates on the premise that complicated functionality should be moved toward the edge of the network with very simple functionality in the core. Edge devices in this architecture are responsible for ensuring that individual user traffic conforms to traffic profiles specified by the network operator and for grouping flows in an aggregated fashion into a small number of classes. Core devices perform differentiated aggregate treatment of these classes based on the marking performed by the edge devices.

This paper describes prototypical implementations of key building blocks required in a Diffserv network. The prototypes are used to study the single queue, dual drop-preference implementation of Assured Services. We explored three different RED models that could be utilized to provide Assured Services. Experimental results are discussed that show how a clear discrimination in bandwidth can be achieved by Assured Services traffic over Best Effort traffic. Finally, we describe tests to study the how real-time UDP flows are affected when utilizing Assured Services in an under-provisioned network.

The organization of the paper is as follows: Section 2 provides an overview of the emerging Differentiated Services architecture. Section 3 examines the most popular proposed services for Diffserv. Section 4 focuses on three RED models that could serve as a basis for Assured Service. Section 5 describes a prototypical implementation of key components in the Diffserv architecture. Section 6 contains the results and analysis of our experimental work with Assured Services in Diffserv. Finally, section 7 summarizes the conclusions of our study and points to areas of future work.

### 2.0 Components of a Differentiated Services IP Network

The Diffserv approach to IP QoS seeks to achieve scalability and reduced complexity by utilizing a packet-marking scheme with handling of traffic aggregates at the core. While the edge of the network may deal with per-flow state information, the core network devices only need to manage and maintain information on a limited number of classes. Edge network devices merge multiple flows into aggregated classes based on administrator specified policy.
There is still much discussion as to what exactly constitutes the Differentiated Services-capable IP Network. The IETF Diffserv working group is currently engaged in defining what such networks will consist of. [1][2] and [12] provide an overview of the Diffserv architecture and framework. Though the exact details of the Diffserv architecture are yet to be finalized, the following could be viewed as the major components that will exist in some form: i) Policy/Network management, ii) edge device functionality, iii) core device functionality.

The first major component in the Diffserv approach is the network policy/resource management infrastructure. The policy management infrastructure allows the network operator to specify QoS network policy – i.e. which traffic gets what kind of service on the network. This high-level policy is translated into device-level policy and downloaded to individual routing devices. A policy management platform will consist of policy servers, policy editors, policy decision points and the policy enforcement points. Methods for distributing policy information in a cohesive consistent manner across the network are currently being discussed. Initially, policy management platforms will include some form of simple resource management. Ideally, dynamic QoS request and allocation of service will be supported by this infrastructure. However, the area of dynamic network resource allocation is one that has many open issues. E.g. since the state of the network is constantly changing, how will this information be relayed to policy managers in a timely and useful manner? Further, there is a requirement to ensure that this kind of control traffic does not consume more than its fair share of network resources. For now, initial realization of Differentiated Services QoS networks will rely on static provisioning and allocation of bandwidth.

The second major Diffserv components are nodes at the edge of the network. These are typically Routers but can also be gateways, nodal bandwidth management devices and possibly trusted end-hosts. These nodes examine incoming packets and classify them according to policy as specified by the network operators. The packets are then marked (in the former TOS field) to reflect the desired level of service. Other functions that might be performed by the edge device include metering, traffic shaping or traffic policing – often referred to as traffic conditioning. To ensure that user traffic profiles are adhered to, edge device responsibility also includes shaping or policing. This allows network operators to control the amount of traffic injected into a network and ensure some measure of end-to-end service differentiation. Traffic shaping is one possible method of restricting user traffic to its operator-defined profile. Traffic policing is the alternative to shaping. Policers are responsible for tracking incoming traffic against user profiles. Out-of-profile packets are either marked specially or discarded. Those packets that are specially marked can always be transmitted to downstream Routers under normal conditions. Those Routers can either transmit the packets onwards or discard them if they experience congestion or provisioning rules are violated.

Figure 1  Sample Network Configuration in the Differentiated Services Architecture

The second major Diffserv components are nodes at the edge of the network. These are typically Routers but can also be gateways, nodal bandwidth management devices and possibly trusted end-hosts. These nodes examine incoming packets and classify them according to policy as specified by the network operators. The packets are then marked (in the former TOS field) to reflect the desired level of service. Other functions that might be performed by the edge device include metering, traffic shaping or traffic policing – often referred to as traffic conditioning. To ensure that user traffic profiles are adhered to, edge device responsibility also includes shaping or policing. This allows network operators to control the amount of traffic injected into a network and ensure some measure of end-to-end service differentiation. Traffic shaping is one possible method of restricting user traffic to its operator-defined profile. Traffic policing is the alternative to shaping. Policers are responsible for tracking incoming traffic against user profiles. Out-of-profile packets are either marked specially or discarded. Those packets that are specially marked can always be transmitted to downstream Routers under normal conditions. Those Routers can either transmit the packets onwards or discard them if they experience congestion or provisioning rules are violated.
The final components in the Diffserv architecture are the Routers at the core of the network. These Routers will have simpler functionality than the edge devices. These devices forward packets with certain PHB (Per Hop Behaviour) depending on the particular packet marking. PHB is the Diffserv architecture term used to describe the forwarding treatment accorded a packet at core network devices. It is the externally observable forwarding behaviour applied against the packet at a particular hop. This observed behaviour is due to treatment received from a particular queuing discipline such as Priority Queueing, Weighted Fair Queueing, RED etc. Different packet markings will invoke different PHBs. Combinations of different PHBs and policy rules applied end-to-end result in a realized service as viewed by the user. Many methods, stateful and/or stateless may be used at the edge to decide on the packet marking required but the core network only has to inspect one fixed header field on a packet by packet basis in order to determine the forwarding treatment.

Network solutions for offering and guaranteeing QoS will vary according to the needs and requirements of each service provider. The above components are building blocks that will be combined to deliver some form of QoS relevant to particular networks. Different networks will utilize only those building blocks that they require. Thus, for example, some organizations might want to perform classification and marking on their egress nodes while others might want to do it on their ingress nodes while others might want to do it on both.

Figure 1 illustrates a sample scenario where the various building blocks can be utilized to deliver end-to-end IP QoS. The Figure illustrates how 2 end-hosts would communicate together with their traffic going through a series of edge and core networking devices. In the depicted scenario, the end-host contracts a service with the local policy/resource manager. In the initial deployment of Diffserv devices, this contract will be static. Over time it might evolve to include dynamic QoS requests. The Policy/Resource Manager translates this contract into device-relevant QoS traffic profile and downloads this to the edge device. Edge devices perform classification, marking and policing or shaping based on this policy. Core devices treat packets according to the PHB specified by the packet marking. All of this results in an end-to-end service as viewed by the users.

3.0 Proposed Services and Related Work

The IETF Diffserv Working Group is currently in the process of standardizing certain key elements in the Diffserv architecture. The working group will not standardize the services to be offered in a Diffserv-based network. Instead, it will focus on standardizing certain building blocks such as PHBs. Service Providers will offer end-to-end services by combining PHBs with policy rules.

There have been a number of proposals describing service schemes that realize a Differentiated Services network. Two of the most popular proposals focus on Assured and Premium services.

A description of Premium Service can be found in [7]. This service is intended to provide a low delay, low jitter guarantee and is geared to the needs of real-time end-applications. The key elements in providing such a service include an EF (Expedited Forwarding) PHB and some form of strict shaping at the edge of the network. The EF PHB [17] ensures that packets are serviced at each node with bounded delay. The traffic shapers ensure that traffic is burst-limited and complies with the peak user profile. Packets exceeding the profile limit are either delayed or dropped. The packets are not downgraded in service and forwarded.

The other key service proposal is Assured Service. A description of this kind of service can be found in [8] [19]. The service is intended to allocate the bandwidth of the Internet to different users in a controlled way during periods of congestion. To alleviate congestion and limit delay in individual nodes, packets need to be dropped. The key to Assured Service is that packets marked with high priority are less likely to get dropped during congestion than packets marked with a lower priority. Thus, the higher priority packet is said to have a lower drop probability. The key elements in this service are the AF (Assured Forwarding) PHB [13] (currently being specified in the IETF) and some kind of policer at the edge of the network.
The policer is responsible for metering the flow and gauging whether or not it has exceeded its contracted traffic profile. If it is within its profile, it is marked with a higher priority. If it has exceeded its profile, intelligent marking is performed on the out-of-profile packets to indicate that downstream Routers can treat them with higher drop probability during times of congestion. Intelligent marking is required for applications that utilize adaptive protocols such as TCP. This is because TCP’s performance can be greatly hindered by multiple sequential packet drops [22].

The AF PHB is responsible for enforcing differentiation when a Router experiences congestion. It achieves this by mapping the packet markings to a suitable packet discard algorithm.

The Working Group specification is not intended to include details on how PHBs are to be implemented. It is left to the individual implementers of Diffserv devices to determine what underlying queuing mechanisms will be used to realize the PHBs. Thus for example, the EF PHB could be realized by a priority queuing mechanism where the EF-marked packets are placed in the highest priority queue. Alternatively, the same PHB could be implemented using WFQ (Weighted Fair Queuing) where again, EF has the highest weight. One possible implementation of the AF PHB is to use a single queue FIFO employing a RIO-like algorithm as proposed in [8]. Another possible implementation is to utilize multiple priority queues with multiple different levels of drop preference within each queue – as hinted by [13]. Regardless of the specific implementation approach selected for the AF PHB, the packet discard algorithm will likely be based on the RED (Random Early Detect) buffer management technique [14].

The concept of an Assured Service is appealing as it proposes simple mark and drop mechanisms to realize QoS. It is important to understand exactly what kind of end-to-end service could be created for an end user using this service. Even more important is to understand the types of applications that could utilize this service and if any form of quantifiable guarantees can be provided by this service. There have been a number of papers [10][18][19] discussing simulation results with Assured Service.

Clark and Fang [19] describe simulation results that show how an Assured Services-based infrastructure can be realized with incremental changes in the architecture of today’s Internet. Another study by Ibanez and Nichols [10] does more detailed simulations of the Assured Service under a variety of situations and concludes that this service cannot provide “clearly defined and consistent rate guarantees”. The authors recommend that further work is required to determine an appropriate use of the Assured Service. Finally, using simulation, Kim and Thomson [18] study different issues associated with active queue management techniques for providing differential levels of service.

In this paper, we use prototypical implementations of the Diffserv components to study the single queue, dual-drop preference implementation as a basis for the AF PHB. The next section discusses different possible ways in which RED can serve as the packet discard algorithm for the AF PHB.

4.0 RED Models for the AF PHB

A RED-like algorithm is the most likely basis for the packet discard mechanism used in the AF PHB. RED is an active queue management technique that is designed to detect upcoming congestion and provide back-pressure to adaptive applications by dropping their packets. It operates based on the average queue size. When the queue size is below a minimum threshold (\(\text{min}_b\)), RED admits all packets. When the queue is above a maximum threshold (\(\text{max}_b\)), RED drops all packets. When the average queue size is between \(\text{min}_b\) and \(\text{max}_b\), RED drops packets with an ever-increasing probability up to \(\text{max}_p\).

To achieve differentiation between different priority classes, different sets of RED parameter values would need to be maintained for each class. Thus, if there are two priority classes, two sets of RED parameters would be required. Each set would have a different impact on arriving packets in its class based on its own unique RED parameters. The RIO algorithm [19] specifies one algorithm for maintaining multiple RED parameters.
In the following sections, we describe experiments that utilize a similar but different algorithm than RIO. The tests experiment with one class of Assured Service and the Best Effort Class. Thus, the AF PHB needs to differentiate between only two classes of traffic. In RIO, the decision of whether or not to drop out-of-profile packet is taken based on the number of total packets (this includes in-profile) in the system. The decision to drop in-profile packets is based on the number of in-profile packets.

In our experiments, we decouple the decision to drop Best-Effort (BE) packets from any information on the total number of packets in the system. Instead, the decision to drop Assured Services (AS) packets is based solely on the number of AS packets in the system and the decision to drop Best Effort packets is based solely on the number of Best Effort packets in the system.

Although we acknowledge the relationship between in-profile and out-of-profile packets and thus, the motivation for the RIO algorithm, we were interested in studying the benefits of decoupling the two decisions. The reason for this is: (i) It is easier to implement (ii) It appears as though it would be easier to extend this algorithm to a multiple queue, multiple drop (more than 2) preference model (iii) Decoupling ensures that congestion due to disproportionate AS traffic is not resolved at the expense of BE traffic. The implication of this principle is that it does not allow bandwidth borrowing between the two classes.

The next task is to set the key parameters that the RED algorithm revolves around. Based on recommendations in [14], we set \( w_q \) to 0.002. The setting of this variable determines how closely the average queue size tracks fluctuations in the real queue size. To simplify the choice of parameter selection, we choose \( \max_p \) for BE such that it has a higher value than \( \max_p \) for AS. This is done independently of the choice of the remaining two parameters.

In the experiments described in subsequent sections, \( \max_p \) values of 0.02 and 0.1 were used for AS and BE packets respectively. This decision was based on the fact that the Internet appears to be experiencing 5% packet drop rate that can mostly be attributed to congestion. Thus, if we spread out the packet loss rate amongst the AS and BE traffic assuming a 50/50 ratio of traffic levels, we obtain an overall drop rate that is slightly higher than average congestion seen in the Internet today.

In setting \( \max_{th} \) and \( \min_{th} \) for AS and BE, there are three different possibilities as captured in Figure 2. One possibility is shown in Figure 2a. In this example, \( \max_{th} \) and \( \min_{th} \) are chosen to be the same for both AS and BE. Thus, differentiation is achieved based strictly on their respective \( \max_p \) values.

Another possibility is shown in Figure 2c. In this example, the drop probability regions for BE and AS do not overlap, i.e. \( \min_{th} \) for AS is equal to or greater than \( \max_{th} \) for BE. Such a configuration ensures that, if we assume equal AS/BE load, AS packets will only start being dropped after 100% of BE packets are being dropped. It also places a hard limit on the amount of delay that AS packets can tolerate due to presence of BE packets in the queue.

The final possibility for parameter setting is as reflected in Figure 2b. In this Figure, there is an overlap between the \( \min_{th} \)–\( \max_{th} \) ranges for BE and AS, i.e. \( \max_{th} \) for BE is greater than \( \min_{th} \) for AS but less than...
maxₐ for AS. This setting causes another type of behaviour under equal-load scenario where BE packets are the first to start getting dropped. If the average amount of both BE and AS packets in the queue keep growing, AS packets will also start getting dropped although at a slower rate than the BE packets.

Experimentation was carried out to understand the dynamics of RED operation using the three possibilities discussed above. A discussion of the results can be found in Section 6.0.

5.0 Implementation of Diffserv Components

Alpha working prototypes were developed for all the key components in the Diffserv architecture. This includes an edge device, a core device and rudimentary policy management capability.

The core and edge device functionality was implemented on a Pentium-200 MHz hardware platform running VxWorks as the RTOS (Real Time Operating System). The current implementation utilizes a 4-port OSICOM PCI Ethernet card. One port is used for network management and downloading of policies to the devices. The other three ports are live data ports that are connected to subnets. The devices include Mombasa [9] – a lightweight embedded web server that is used to facilitate device network management. With the inclusion of Mombasa, network administrators are able to manage the edge and core devices using any standard HTTP web browser. e.g. Netscape or Internet Explorer. Among other things, Mombasa enables such features as remote policy specification, remote device configuration management, remote setting of QoS configurable parameters, remote device reboot and detailed statistics monitoring.

The software was written in a modular fashion to allow as much code reuse as possible. The same software load runs on both the core and edge devices. The user via a standard web browser is able to set the network element to operate as either an edge or core device. Certain functionality such as queue service/management algorithms and forwarding mechanisms is common to both devices. Classification, packet marking, traffic conditioning and any per-flow management related work is limited to the edge device.

The software was written such that it provided a configurable set of basic QoS-relevant mechanisms or building blocks that the administrator configures to realize the necessary Diffserv capability. Examples of the configurable mechanisms include:

- Number of queues
- Queuing service discipline. E.g single queue FIFO, Multi-queue Strict Priority etc.
- Selection of a buffer management algorithm. E.g Drop-Tail, RED
- Number of RED drop preferences per queue with ability to specify minₐ, maxₐ, maxₚ and wₐ on a per drop-preference basis
- Mapping of code-point to queue number and RED drop preference
- Specification of physical ports on which edge device functionality is to be activated
- Policy specification – this allows a remote system administrator to specify that certain source IP addresses should get prioritized treatment
- Classification
The network administrator, using a web browser as the policy editor, can set all of the above mechanisms. There is no central policy manager so the operator must download parameters to each device directly. There is also no explicit signaling between the end-host and any network nodes or the browser to request service.

The edge device functionality implemented to date includes classification, marking and metering. Figure 3 shows a model of the edge device functionality. Classification is based on Layer 3 source IP address. Packet marking is performed in conformance with [12], where the 6 most significant bits of the former TOS field are used to represent the PHB to be applied against the packet. At this point, the devices have a one-to-one mapping between PHBs and service classes.

### 6.0 Experimental Description and Results

The experimental setup utilized can be seen in Figure 4. Three prototypical devices were utilized to represent 2 edge and 1 core network element. The Netperf [20] tool was used to generate TCP and UDP flows with specified packet sizes and duration. The rest of the setup comprised Pentium 90MHz, 133MHz, 200MHz and 400MHZ host computers running FreeBSD 2.2.6 and Linux 2.0.34 as the operating systems.

![Figure 4 Experimental Setup](image)

The underlying basis of the experiments was to study certain issues associated with the single queue, dual drop-preference model for realizing Assured Services. We note that [19] presents results showing how a single-queue, dual drop preference Assured Services infrastructure can be used to provide differentiated service to applications using TCP flows. The benefits of this implementation are that it is rather simple to realize. The authors state that this single queue model can be utilized as a basis for a Differentiated Services infrastructure that supports both real-time and non-real time applications.

A key requirement for the correct functioning of the infrastructure described in [8] is sufficient over-provisioning of network resources. However, over-provisioning is extremely difficult to achieve in the Internet and it is likely that the network will be under-provisioned. We were interested in seeing what kind of service assurance can be provided in such a network and further, whether or not real-time applications can be supported under such conditions.

The experiments were performed such that the edge devices classified and marked traffic appropriately. Congestion was created at the core. Since the goal was to study the under-provisioned case, per-flow traffic policing was not performed at the network edge.
6.1 Experiment 1

The goal of the first experiment was to explore how selection of RED parameters impacted the service gained by AS traffic over BE traffic. In particular, we studied setting of RED parameters \( \text{min}_{th} \) and \( \text{max}_{th} \) using the different models described in Figure 2.

Netperf was used to generate all the flows in this experiment. Equal numbers of AS and BE TCP flows were caused to pass through the core device. BE flows originate in Linux3 and are terminated in Linux4. AS flows originate in Linux2, are marked and classified by the first edge device, and terminate in Linux5. Acks between Linux5 and Linux2 are also marked and classified as AS traffic by the second edge device.

Three different tests were carried out. Each test was run for 200 seconds and was repeated with 3, 5 and 7 TCP flows per class. RED was implemented using packet counting as opposed byte counting. Table 1 summarizes the \( \text{max}_{th} \)-\( \text{min}_{th} \) parameter choices for the three different RED models illustrated in Figure 2.

<table>
<thead>
<tr>
<th>RED Model</th>
<th>( \text{AS min}_{th} ) (# packets)</th>
<th>( \text{AS max}_{th} ) (# packets)</th>
<th>( \text{BE min}_{th} ) (# packets)</th>
<th>( \text{BE max}_{th} ) (# packets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>30</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>40</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>50</td>
<td>10</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 2 shows the drop packet count and exponentially smoothed average queue size for the three RED models with a variable number of flows per class of traffic. To analyze the results, we examine the case for 7 flows per class. We observe that model (a) has a large number of packet drops for both BE and AS traffic. AS packet drops are relatively high because the smoothed average queue size is close to \( \text{max}_{th} \) and so more packets tend to get dropped. On the other hand, models (b) and (c) show a low drop count for the AS class since the AS \( \text{min}_{th} \) is higher in these models. A consequence of higher thresholds is higher average queue sizes.

<table>
<thead>
<tr>
<th>Figure 2</th>
<th>( \text{AS min}_{th} ) (# packets)</th>
<th>( \text{AS max}_{th} ) (# packets)</th>
<th>( \text{BE min}_{th} ) (# packets)</th>
<th>( \text{BE max}_{th} ) (# packets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model a</td>
<td>314</td>
<td>26</td>
<td>1059</td>
<td>264</td>
</tr>
<tr>
<td>Avg Q Size</td>
<td>13.4</td>
<td>14.8</td>
<td>17.3</td>
<td>24.3</td>
</tr>
<tr>
<td>Model b</td>
<td>327</td>
<td>0</td>
<td>974</td>
<td>83</td>
</tr>
<tr>
<td>Avg Q Size</td>
<td>13.3</td>
<td>14.7</td>
<td>17.2</td>
<td>26.8</td>
</tr>
<tr>
<td>Model c</td>
<td>299</td>
<td>0</td>
<td>958</td>
<td>0</td>
</tr>
<tr>
<td>Avg Q Size</td>
<td>13.4</td>
<td>14.8</td>
<td>17.4</td>
<td>26.9</td>
</tr>
</tbody>
</table>

Since the decision to drop AS and BE packets are based solely on their respective number of packets in the queue, we see a consistent smoothed average queue size for BE. This indicates that regardless of model choices, the decoupling of BE drop decision from number of AS packets in the queue, allows BE to get a fair share of the bandwidth. Of course, it is debatable as to whether or not BE’s share of the bandwidth needs to be protected.

6.2 Experiment 2

The goal of the second experiment was to study whether a single queue with dual-drop preference could provide sufficient service discrimination to serve as a basis for Assured Service. We observe average queue length, number of packet drops and aggregate bandwidth utilization for AS and BE traffic as the number of competing flows increases.
Figure 5: Average Queue Occupancy: Assured vs Best Effort Classes

Figure 6: Drop Packet Count: Assured vs Best Effort Classes

Figure 7: Bandwidth Utilization: Assured vs Best Effort Classes

Figure 8: Histogram plot of UDP packet inter-arrival times for 64Kbps CBR Stream
BE flows originated from Linux3 and terminated in Linux4. AS flows originated in Linux2 and terminated in the Linux5 host. Equal numbers of flows were run for each type of traffic. The number of flows in the system started at 2 BE and 2 AS flows and was gradually raised to 9AS and 9BE flows.

The results of the experiments can be seen in the Figures of the previous page. Figure 5 shows the average number of packets for each class present in the queue as the number of flows is increased. The general trend reflected in the graph is diverging queue occupancy rates for the two classes of traffic with the average number of AS packets in the queue being larger than BE’s packets. There is obviously a clear differentiation achieved. AS has a queue occupancy gradient that is larger than that of BE. This is because BE starts to drop traffic early and drops more aggressively than AS. The average queue occupancy for both AS and BE are the same until the min\_th mark for BE is crossed. At this point, BE traffic starts getting dropped and then begins to approach max\_th in an asymptotic fashion. The AS average queue occupancy on the other hand, grows steadily until it is halfway between min\_th and max\_th where it starts leveling off.

The average queue occupancy for either class has an impact on the number of dropped packets. Figure 6 captures the drop packet count per traffic class as the number of flows are increased. Since the average queue occupancy rises slowly with each added flow, a proportional number of packets are dropped. The AS curve shows a similar linear behaviour up to a load of 6+6 flows. At this point, the drop packet count increases rapidly since the average queue occupancy oscillates and frequently hits max\_th – see Figure 9 for a detailed trace of the average queue occupancy of AS and BE packets in the 9+9 flow scenario.

In the final analysis, the end user evaluates service discrimination by delay, bandwidth, delay jitter etc. We choose to observe bandwidth to determine what kind of differentiation can be achieved by having different RED parameters for AS and BE. The results of this test are captured in Figure 7 With two flows per class, we observe that BE acquires more bandwidth than AS. This is because the BE traffic has a lower RTT (Round-Trip Time) and there isn’t sufficient bottleneck to cause packet drops in either class. As the number of flows increases, the bandwidth share of AS clearly exceeds that of BE. This is due to AS having fewer packet drops. However, after 7+7 flows, since AS queue occupancy approaches max\_th, we observe a commensurate leveling off in bandwidth gain. In fact, beyond a point, the AS performance gain actually takes a downturn.

One clear conclusion from the above is that with proper choice of RED model and parameters, distinct service differentiation can be achieved. Another conclusion is that the bandwidth usage of AS flows is determined by a large number of factors. Prediction of bandwidth usage and guarantee will prove challenging and requires future investigation.

**Figure 9: Average Queue Occupancy with 9 AS and 9 BE Flows**
6.3 Experiment 3

The goal of the final experiment is to observe the kind of service that could be obtained by real-time AS UDP flows even though they shared a single queue with multiple AS and BE TCP flows. This has implications on whether or not such an implementation model can be utilized in a network where Voice-over-IP users desire certain audio quality.

We initially started experimenting with voice-over-IP using popular software and cards. It was noticed that reasonable voice quality can be obtained with around 5-6Kbps. However, quantification of voice improvement proved difficult as the voice applications are based on RTP (Real-Time Protocol) and are adaptive. On detecting congestion, RTCP (RTP Control Protocol) messages provide notification of audio quality. The application sender appeared to adjust its traffic sending pattern based on the feedback of these messages. As such, instead of generating a relatively unimodal distribution of packet inter-send times, we observed an exponentially decreasing distribution of the inter-sending times. Further, the nature of the traffic stream generated by the source appeared to change based on the congestion level which itself was constantly changing. Quantification of results and comparison of packet inter-arrival times at the receiver proved a very challenging task without details about the packet sending algorithm and in the presence of constantly changing source stream pattern.

Instead, we chose to use a 64Kbps CBR UDP stream to mimic a a stringent example of an audio stream. The test was carried out with 8 AS TCP flows, 8 BE TCP flows. The UDP flow was classified as AS traffic. A negligible amount of packet drops were observed for the UDP flow. This is as expected due to the low bandwidth usage of the flow. If packet drops for such flows remain at such levels then audio/voice applications will have no problems with packet drops as they appear to be resilient to up to 5-10% packet loss.

Next, we observed and studied packet inter-arrival times at the receiver. As shown in Figure 5, the total queue size for AS and BE packets approached 55. Thus, the maximum inter-arrival delay variation that was expected is around 70ms (Pkt Queue Len X Pkt Transfer Time for 10BT). Examining a history of inter-arrivals in Figure 8, we see a maximum spread that is below the estimated maximum delay variation. This kind of delay variation appears to be within the acceptable limits for voice applications. However, we note that the buffer size and RED parameters used were deliberately small. In a situation with larger delay-bandwidth products as required by TCP flows, larger buffers might be needed. This would greatly affect the inter-arrival delay variation and would impact perceptible voice quality. Resolving this problem might necessitate moving to a multiple queue model. Even in such model, there is still the question of whether or not TCP and UDP AS flows can coexist given the fact that 1 delay-bandwidth product of buffer might be required to support TCP bursts.

7.0 Conclusions and Future Work

This paper explored issues associated with providing for an Assured Service in a Differentiated Services network. In particular we analyze the single queue, dual drop-preference implementation of Assured Service. Experiments were carried out using prototype reference implementations in scenarios intended to resemble an under-provisioned network that experiences congestion. A RED model with over-lapped minth and maxth ranges for AS and BE was experimented with and analyzed. It appeared to provide a moderate tradeoff between average queue size and packet drop count rate. We have verified that bandwidth differentiation between Assured and Best Effort packets can be achieved. However, we are unsure about the extent to which bandwidth differentiation can be predicted. The results also appear to show that real-time applications such as voice can achieve desired quality in a single queue with limited buffer size. However, this may have implications on the performance of TCP Assured flows that share the queue. As such, it may be necessary to consider a multiple queue model where some kind of protection is accorded the real-time UDP flows.
Future plans include detailed implementation and study of service differentiation achieved with multiple queues and multiple drop preferences. A further extension of this is to separate the TCP and UDP flows into separate queues. It is also interesting to study how ECN [15] and BECN [16] can be used for improved service differentiation with fewer packet drops.

8.0 Acknowledgements

We are grateful to Don Bennet, Angelis Koutsos and Steve Jaworski who assisted with various lab environment and equipment issues that arose during our work.

9.0 References


