In IEEE International Conference on Telecommunications 2001

# Best Effort Differentiated Services: Trade-off Service Differentiation for Elastic Applications

Victor Firoiu
Advanced Technology
Nortel Networks
600 Tech Park
Billerica, MA 01821 USA
vfiroiu@nortelnetworks.com

Xiaohui Zhang
Advanced Technology
Nortel Networks
600 Tech Park
Billerica, MA 01821 USA
xiaohui@nortelnetworks.com

Yang Guo *
Department of Computer Science
Univ. of Massachusetts
Amherst, MA 01003 USA
yguo@cs.umass.edu

## Abstract

We propose an architecture and several mechanisms for providing "Best Effort Differentiated Services" (BEDS), a set of services similar to Best Effort in that the packet delay and drop probability depend on the network conditions, but with the added feature of providing a tradeoff diferentiation in delay and drop probability between the services in this set.

This proposal starts from the observation that a single Best Effort service does not fit the needs of all types of elastic applications. In our proposal, the "loss-conservative" service has smaller packet loss probability but larger delay than the "delay-conservative" one. The former is suited for file transfer applications, whereas VoIP can significantly benefit from the delay-conservative service.

We provide experimental results that confirm our models' predictions that file and web transfer and interactive voice applications can simultaneously benefit from this set of services. The experiments were conducted using networks of Nortel routers and switches implementing our proposed mechanisms.

## I Introduction

In the last 15 years we have witnessed a sustained exponential growth of the Internet in both the number of connected hosts and the capacity of its data links. The "traditional" e-mail and file transfer applications, that accounted for more than 90% of traffic in the first 10 years, have largely been surpassed by the Web (HTTP) and multimedia (voice, music and video) traffic [6]. Still, most of the traffic service offered commercially has been of "Best Effort" type, whereby data packets are forwarded at the network layer with no guarantee or preference for reliability or timeliness of delivery. This service, supplemented by the reliable transport protocol TCP is a reasonably good service for most file transfer applications, from e-mail to Web browsing, but is a difficult environment for delay-sensitive applications such as interactive voice conversations (or "Voice over IP", VoIP).

The problem of providing services with specific characteristics such as Quality of Service (QoS) guarantees along with Best Effort service in the same network has been the object of extensive research for more than 10 years. An early protocol for bandwidth reservation, ST-II [7], was followed by another, RSVP [4], coupled with a set of service definitions commonly known as Integrated Services (IntServ) [31, 34]. In both cases, the Quality of Service is provided following a user application request by reserving a part of available bandwidth of each data link traversed by the application's traffic (also known as a flow). Although these protocols proved to provide QoS reliably in experiments, they have seldom been deployed in production networks due to scalability problems. Keeping per-flow state information and performing per-flow packet treatment at a link traversed by millions of simultanous flows is considered too complex.

In 1998, an alternative, Differentiated Services (DiffServ) [2, 21, 15], was proposed, whereby flows with similar requirements are aggregated in a class, and resource reservation and differentiated packet treatment is performed per-class instead of per-flow. This aleviates the scaling problem of RSVP/IntServ, but introduces others, such as the need for a system for network-wide bandwidth management (or bandwidth brokers) and the difficulty of providing per-flow QoS in the context of aggregation [14]. Moreover, other significant overhead such as admission control and authentication is required by both RSVP/IntServ and DiffServ [1].

The absence of services with specific characteristics has nevertheless stimulated the development of elastic applications. Popular VoIP applications such as Vocaltec's IPhone and Microsfot's NetMeeting provide reasonable voice quality over Best Effort, but their end-to-end delay is far from acceptable. This delay is in large part due to large packet queues accumulated at congested links. Traditionally, large buffer capacities have been configured for Best Effort

*This work was done when Yang Guo was working as a summer intern at Nortel Networks.

[1] DiffServ, as defined by [2, 21, 15] does not have such requirements since it does not define services, but any service based on DiffServ needs some form of access control

in order to minimize packet losses during congestion, a setup mainly suited for file transfer applications and not for time-sensitive applications.

In this paper, we propose an architecture and several mechanisms for providing "Best Effort Differentiated Services" (BEDS), a set of services similar to Best Effort in that the QoS provided depends on the network conditions, but differentiated in their tradeoff between packet delay and packet loss probability. This proposal starts from the observation that one Best Effort service does not fit the needs of all types of elastic applications. In our proposal, the "loss-conservative" service has smaller packet loss probability but larger delay than the "delay-conservative" one. The former is suited for file transfer applications, whereas VoIP can significantly benefit from the delay-conservative service. To realize this architecture, we propose several simple mechanisms that include variants of Weighted Fair Queuing (WFQ) combined with Random Early Detection (RED) active queue management. Therefore, our proposed mechanisms have the same low computational requirements as the known implementations of WFQ and RED.

Our proposal for Best Effort Differentiated Services provides a low-overhead way to introduce differentiated services in the Internet. It enables a range of elastic applications with heterogeneous requirements to be evenly serviced without the overhead of per-flow state and per-hop packet treatment, admission control or bandwidth broker. Moreover, BEDS is not required to be implemented in all the network elements on a flow's path, but the more routers implement it, the more effective is the service differentiation. Therefore, BEDS can be incrementally deployed in the network, and can be a first step in deploying Differentiated Services.

Service differentiation without guarantees has been the object of several recent contributions and our proposal differs from them in some key aspects. Asymetric Best Effort [16] includes two service classes with high throughput and low delay respectively. ABE is based on the assumption that all serviced traffic is TCP-friendly, i.e., uses the TCP congestion control mechanism. Unlike ABE, our proposal does not have this restriction. Proportional Differentiated Services [8] provides service classes with delay differentiation proportional to a set of parameters, but unlike our proposal, requires admission control. Flow Aware Networking [30] provides service classes with relative delay differentiation and also requires admission control.

The rest of the paper is structured as follows. In Section II we present a model for the performance (packet delay and loss) of TCP and UDP flows traversing congested links. In Section III we examine the impact of service performance on the perceived quality of web transfer and interactive voice applications. In Section IV we present an architecture of BEDS in the context of a known and stationary traffic mix, and present its performance advantages compared to traditional Best Effort. In Section V we describe several mechanisms for BEDS in the context of a dynamic traffic mix. We conclude in Section VI.

# II A model for TCP and UDP traffic dynamics

In this section, we present models for the average throughput, delay and loss probability of FTP/TCP and Voice/UDP flows traversing a congested link. These models provide the context for evaluating the impact of network performance on interactive web and voice applications, and form the basis for our proposed BEDS architecture.

## A *A model for TCP traffic*

Let us first consider file transfer applications over TCP. In the context of a Best Effort network, TCP adjusts its sending rate according to network capacity and traffic load. Recently, several studies including [22, 29, 5] have proposed increasingly accurate models for the performance of TCP as a function of network and end-host conditions. In Appendix we summarize the model proposed in [29], which we will use extensively in this work. According to it, the average throughput of a TCP flow is modeled as a function $T(p, R)$ of network packet drop probability $p$ and of round trip time $R$ (equation (8) in Appendix).

In this work, we consider that network elements (routers and switches) implement Random Early Detection (RED) [12] as queue management algorithm. RED is increasingly popular, being recommended by the IETF [9]. Also, RED, compared to the traditional Tail-drop mechanism, provides more flexibility in choosing network behavior during congestion, as we will see in the following. Essentially, this mechanism drops packets from the transmission queue of a link with a probability $p$ computed as an increasing function of a running average of that queue's size $q$, $p = H(q)$. Obviously, the congestion control mechanism of TCP interacts with the RED algorithm, and analytical models have been proposed recently in [10, 25, 11]. Following [10], in a simple setting with $n$ identical TCP flows having round trip propagation delay $R_0$ and traversing one congested link of capacity $c$, the average throughput of a TCP flow is $c/n$, and the average queue size $q$ and packet loss probability $p$ are the solutions to the system: [2]

$$T(p, R_0 + q/c) = c/n \quad (1)$$

$$p = H(q) \quad (2)$$

where $T(p, R)$ is given in (8). We observe that $T$ is strictly monotone in $R$ and thus is inversible in $R$. We denote by $T^{-1}(p, t)$ the inverse of $T(p, R)$ in $R$

[2] A model for the general setting with heterogeneous TCP flows, propagation delays and multiple congested links is proposed in [11], but does not have a closed form expression. In this paper we consider the simple setting for ease of exposition, similar results are obtained in the general setting.

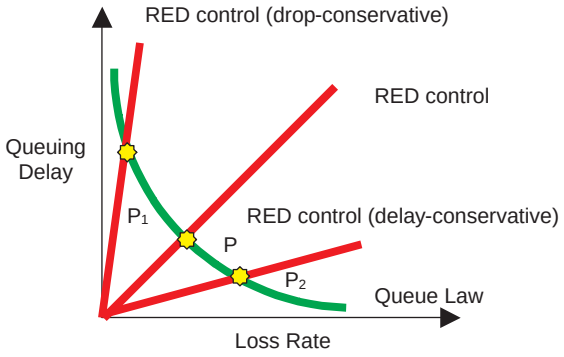


Figure 1: Queue Law function and RED Control Function for TCP traffic

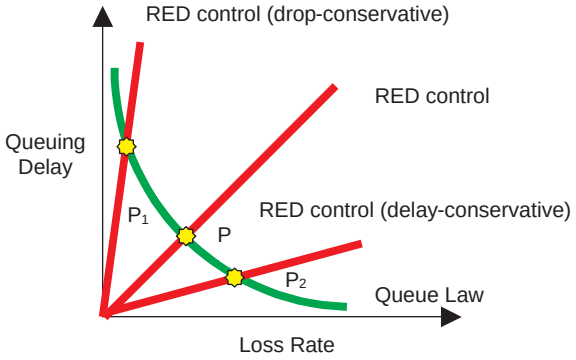


Figure 2: Queue Law function and RED Control Function for Mixed traffic

(i.e., $T^{-1}(p, T(p,R)) = R$), and thus the solution of (1) is $q = c(T^{-1}(p, c/n) - R_0)$. We denote $G(p) = c(T^{-1}(p, c/n) - R_0)$ and name it the "queue law" of our system that becomes

$$q = G(p) \quad (3)$$
$$p = H(q) \quad (4)$$

In Figure 1, the intersection $P$ of the queue law function $G$ and RED control function $H$ is the unique solution of the above system and represents the average operating point (average queue delay and packet drop probability) of the traffic. In other words, the TCP traffic experiences on average the loss rate and queuing delay given by $P$. It also follows that a given traffic set has a unique queue law function, and therefore, for different RED control functions, produces different operating points along that queue law function. We can thus define a range of queue management policies such as delay-conservative and drop-conservative. Under the drop-conservative policy, the traffic has a small loss probability, but a relatively large delay ($P_1$ in Figure 1), whereas under the delay-conservative policy, the traffic has a short delay but a relatively large loss probability ($P_2$);

## B *Voice/UDP traffic*

We consider voice traffic generated by applications such as Microsoft Netmeeting [24] or Vocaltec's IPhone using G.723.1 [17] codec at 6.4Kb/s voice data rate. The Ethernet packet size is 82 Bytes (18 Bytes for Ethernet header and trailer, 20 Bytes for IP header, 8 Bytes for UDP header, 12 Bytes for RTP header and 24 Bytes for voice data), and the Ethernet transmission rate is 21.866Kb/s (or 1 packet/30ms). We model this traffic as an ON-OFF Markov process with average ON time of 3 seconds and average OFF time of 6 seconds.

The voice data is usually carried over UDP instead of TCP because packet retransmission (in case of loss) produces unacceptable delays. Since UDP does not have congestion control, the average queue size produced by a set of Voice/UDP flows is unbounded when the average aggregate sending rate of a set of Voice/UDP flows is larger than the link bandwidth. If the average rate of Voice/UDP traffic is smaller than the link bandwidth, the queuing delay is generally small.

## C *A model for TCP and Voice/UDP traffic*

We consider now a set of $n$ idential TCP flows and $m$ identical Voice/UDP flows traversing a congested link using RED. We model this system by assuming that all flows experience the same average loss probability $p$ and same average queuing delay $q/c$. Denoting by $T$ the average throughput of a TCP flow and by $U$ the average sending rate of a UDP flow and following a similar argument as in Section A, we have

$$nT(p, R_0 + q/c) + m(1-p)U = c$$
$$p = H(q)$$

It follows that the mixed traffic has a queue law similar to the one described in Section A. Moreover, it is equal to the queue law of a system having the same set of TCP flows and whose line rate is reduced by the aggregate average throughput of the Voice/UDP traffic. Figure 2 depicts the queue law function of the mix traffic.

## D *Simulation of the traffic models*

To verify our models, we perform simulation experiments using the NS-2 [23] network simulator. Figure 3 shows the network topology. There are 80 senders ($S_1$, $S_2$, ..., $S_{80}$), 80 receivers ($D_1$, $D_2$, ..., $D_{80}$), and two routers ($R_1$, $R_2$). All the senders are connected with $R_1$ router and all the receivers are connected with $R_2$ router. All the links are 100Mbps except for the link between two routers, which can be set to different values. We use RED as the queue management policy in both routers.

In a first set of experiments, the senders send only FTP/TCP traffic. The bottleneck link bandwidth is set to 9Mbps. There are 80 FTP connections between senders and receivers. The packet size is 1KB.

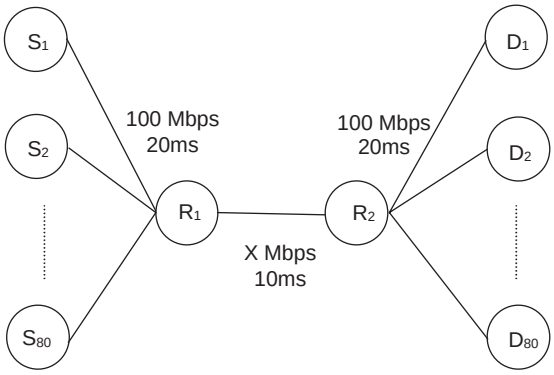


Figure 3: Simulation Network Topology

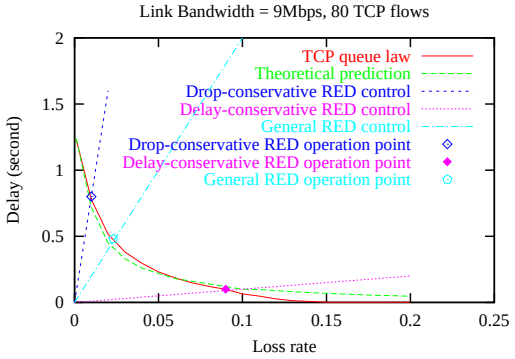


Figure 4: TCP Queue Law

In each experiment in this set, packets are dropped at link $R_1 - R_2$ with a fixed probability taken from the range 0% to 20%, and we measure the queuing delay in $R_1$. The simulations runs for 100 seconds. As illustrated in Figure 4, the measured queue law is very close to our theoretical queue law. We also tested three RED control policies, and we see in Figure 4 that all operating points are close to the intersection of the the queue law function with corresponding RED control function.

In other set of simulations, we set the bottleneck link bandwidth equal to 10Mbps, and generate traffic as a combination of TCP traffic as in the previous experiment, and VoIP/UDP traffic following an On-OFF Markov model described above and with an average rate of 0.8Mb/s. We find the queue law for this mix traffic to be close to the queue law for TCP only traffic in 9Mbps link, as seen in Figure 5.

## E *Test-bed experiment*

We also set up a test bed as in Figure 6 to verify our theoretical model and simulation results. We use four PCs, two senders and two receivers. All the PCs are Intel Pentium II 266MHz with 128 MB memory and equipped with RedHat Linux 6.0 and Linux diffserv patches [33]. The sender PCs are connected with BayStack 450 Ethernet switch [27] and two receiver PCs are connected with BayStack 350

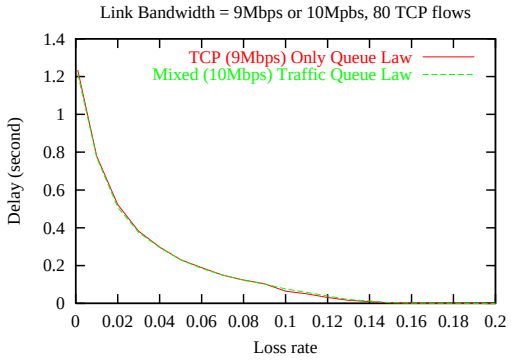


Figure 5: TCP Queue Law vs. Mixed Traffic Queue Law

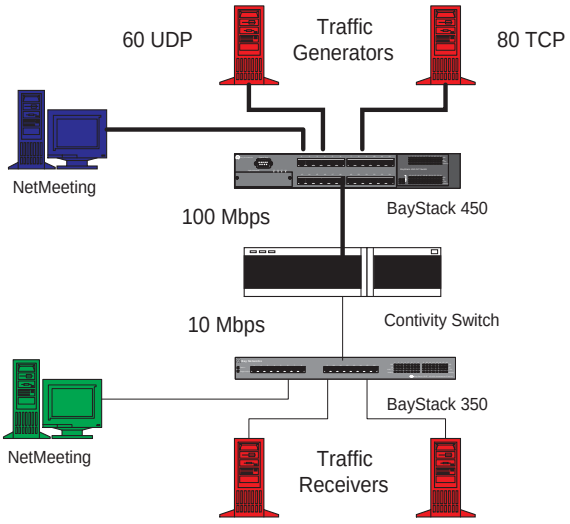


Figure 6: Test Bed Topology

Ethernet switch [26]. Both of the BayStack switches are connected to a Contivity 1500 switch [28]. All the links are configured as full duplex. From BayStack 450 to Contivity, the link bandwidth is 100 Mbps while from BayStack 350 to Contivity, the link bandwidth is 10 Mbps. Thus from sender to receiver, the bandwidth is 10 Mbps. The RED and WFQ queue buffering and scheduling mechanisms have been implemented in the router.

In Figure 7, we show the results of an experiment with 80 FTP/TCP flows and 60 VoIP/UDP flows, the same configuration as our previous ns-2 simulation, except that all other link delays are negligible. We see that both the queue law curve and operating points measured at the test-bed are close to our theoretical prediction.

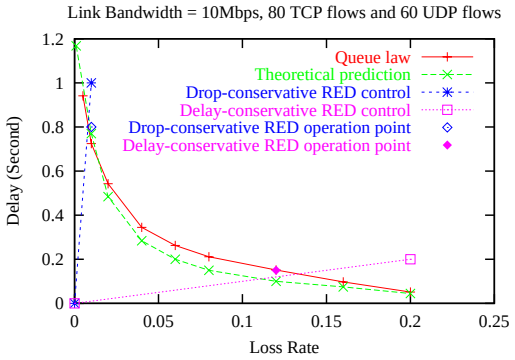


Figure 7: Experiment Result of Nortel Test-Bed

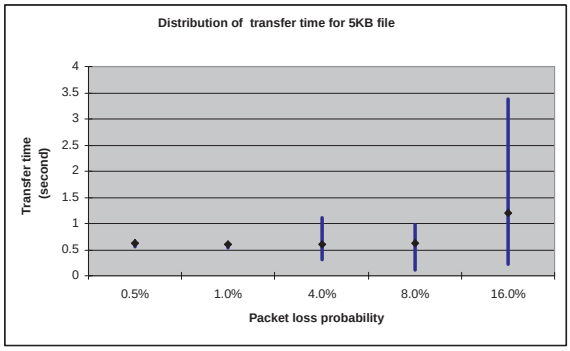


Figure 8: Distribution of transfer time for 5KB files

# III Effect of network conditions on elastic applications

The model for TCP and UDP packet delay and loss probability developed in last section enables us to evaluate their effect on the perceived quality of two of the most used types of applications: file transfer and interactive voice.

## A *File transfer applications*

File transfer, either explicit (FTP) or implicit such as web page download (HTTP) or email (SMTP), constitutes the vast majority of Internet traffic. A set of traffic measurements performed in 1998 in the "very High performance Backbone Service" (vBNS) and reported in [6] suggests that 95% of IP traffic is TCP, of which HTTP is 70%, FTP 5% and SMTP 5%. From the same study, the size of an HTTP session is found to be in the $6..8KB$ range and an FTP session in the $30..240KB$ range. We consider email transfer to be a background (non-interactive) application on which network conditions have a lesser impact, and therefore we do not analize this type of application in this study. In the following we use the model developed in Section A to analyze the effect of packet loss and delay on file transfer applications.

First, we observe that the average time for a file transfer of a given size depends only on the network capacity $c$ and offered load $n$ and not on a particular combination of packet delay and drop probability. This is because the throughput of a TCP flow is independent of the position of its operating point on a given queue law (see Figure 1).

Nevertheless, from the system (1) and (2), we also have that the same throughput $T$ can be achieved in different conditions where queuing delay $d = q/c$ and loss probability $p$ are traded off. This tradeoff can be implemented by choosing different RED control functions $H$. This is depicted in Figure 1, where different control functions $H$ intersect one queue law $G$ in different points.

To study the impact of delay-loss tradeoff on the quality of file transfer, we performed the following experiments. In a first set, we generated 5KB-size file transfers were generated over a 10Mb/s line and we recorded the completion time for each transfer. In each experiment in this set, the RED control function was configured differently, such that the average drop probabilty was $0.005, 0.01, 0.04, 0.08$ and $0.16$ respectively. The results are shown in Figure 8, where the average transfer time is depicted by a dot, and the end-points of each bar represents the 10-th and 90-th percentile of all transfer times in that experiment. We repeated this set of experiments two more times, with file sizes of 10KB and 100KB respectively, and the results are shown in Figures 9 and 10.

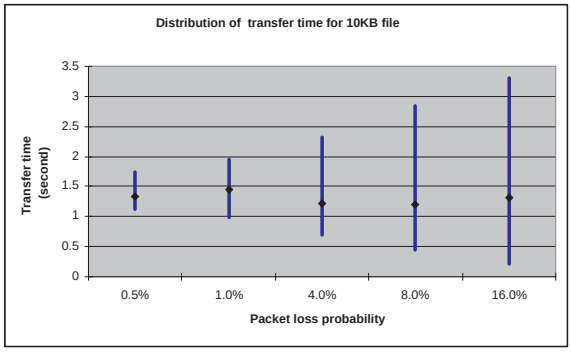


Figure 9: Distribution of transfer time for 10KB files

First, we observe that the average transfer times in all three experiments is almost independent of the loss probability, which confirms our earlier prediction. On the other hand, the variability in file transfer time is large for small file sizes and high loss probability. The explanation is that at each packet loss, a TCP retransmission occurs and sometimes also a TCP timeout. The impact is higher for small transfers for several reasons. First, one retransmission out of a small number of packets in a file has a large impact on the total file transmission time. Also, TCP timeouts are more likely to happen during small file transfers because the TCP congestion window is smaller at the beginning of a transmission ("slow start" phase, see [32] for details).

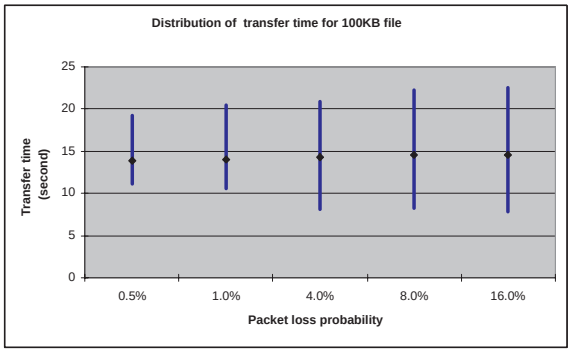


Figure 10: Distribution of transfer time for 100KB files

Several studies such as [1, 3, 13] suggest that delay variation is an important subjective factor in the perceived quality of interactive applications, and that a variation above 10% is not likely to be acceptable. Given that web traffic of 6..8KB size is the vast majority of Internet traffic, and from our simulations in Figures 8 and 9, we conclude that the desirable range of loss probability is below 0.01.

### B *Interactive voice applications*

The impact of network conditions on the perceived quality of voice conversations has been extensively studied. A series of ITU-T publications [20, 18, 19] proposed the "e-model" which models analytically the subjective quality of conversations as a function of many parameters including one-way end-to-end delay, packet loss and type of encoding.

The e-model calculates a transmission rating factor $R$ (not to be confused with the round trip time in the previous section) that represents the perceived quality of a conversation, as a number between 1 and 100. $R = 100$ represents perfect quality, and above 60 is considered acceptable. For the purpose of this work, we simplify the expression for $R$ as follows

$$R = R_0 - I_d - I_e$$

where $R_0$ is the basic quality (with zero delay and loss), $I_d$ is the penalty due to delay and $I_e$ is the penalty due to packet loss and also dependent on the voice encoding scheme. The expression for $I_d$ has analytical but relatively compicated expression, and $I_e$ is given by a set of numbers in a table. Here we do not go in further details, but plot in Figure 11 a set of values for $R$ for a range of delay, loss probability and two encoding schemes, G.711 and G.723.1.

G.723.1 is a preferred encoding for VoIP due to its relatively low bandwidth requirement (6.4Kb/s voice data rate versus 64Kb/s for G.711). For a subjective quality above 60, G.723.1 can sustain up to 4% loss and up to $100ms$ delay. Out of the $100ms$ delay, about $20ms$ are packetization and transmission delay, leaving about $80ms$ for queueing delay. We should note that this model for subjective quality is only informative, and we will use the above numbers only for guidance.

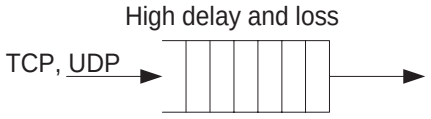


Figure 12: Best Effort

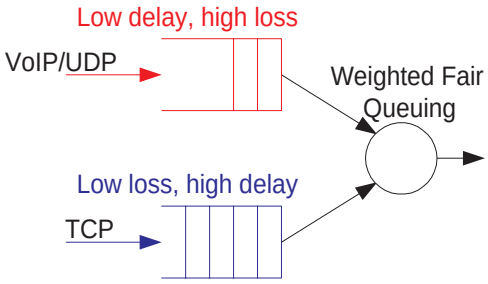


Figure 13: Best Effort Differentiated Services

### C *An aggregate perspective*

Without being an exact study, in this section we have uncovered desirable network conditions for web transfer and voice applications. Web traffic has a good quality when the packet loss is below 1%. On the other hand, VoIP applications have acceptable operation for packet loss below 4% and delay below $50ms$. It follows that the two types of applications are sufficiently elastic to use Best Effort service, but have two different ranges of acceptable operation. In the next sections we describe an architecture for Best Effort services that addresses these different requirements.

## IV BEDS ARCHITECTURE FOR STATIC TRAFFIC MIX

To get the maximum performance for both TCP traffic and UDP traffic, one simple idea is to separate the traffic into different queues. For each queue, the queue management policy can be configured differently. We extend the one RED queue model of Best Effort (Figure 12) into two RED queues serviced according to a Weighted Fair Queuing (WFQ) discipline (Figure 13). Packets are dispatched to the queues according to a Differentiated Services marking. Each RED queue has a separate RED parameter set to control the traffic and a WFQ weight. The RED control function for one queue is configured as lower loss rate but higher delay for TCP traffic, and the other queue is configured as lower delay but higher loss for UDP traffic.

We use the same traffic as Figure 4, 80 TCP flows and 60 UDP flows. TCP traffic is scheduled to a

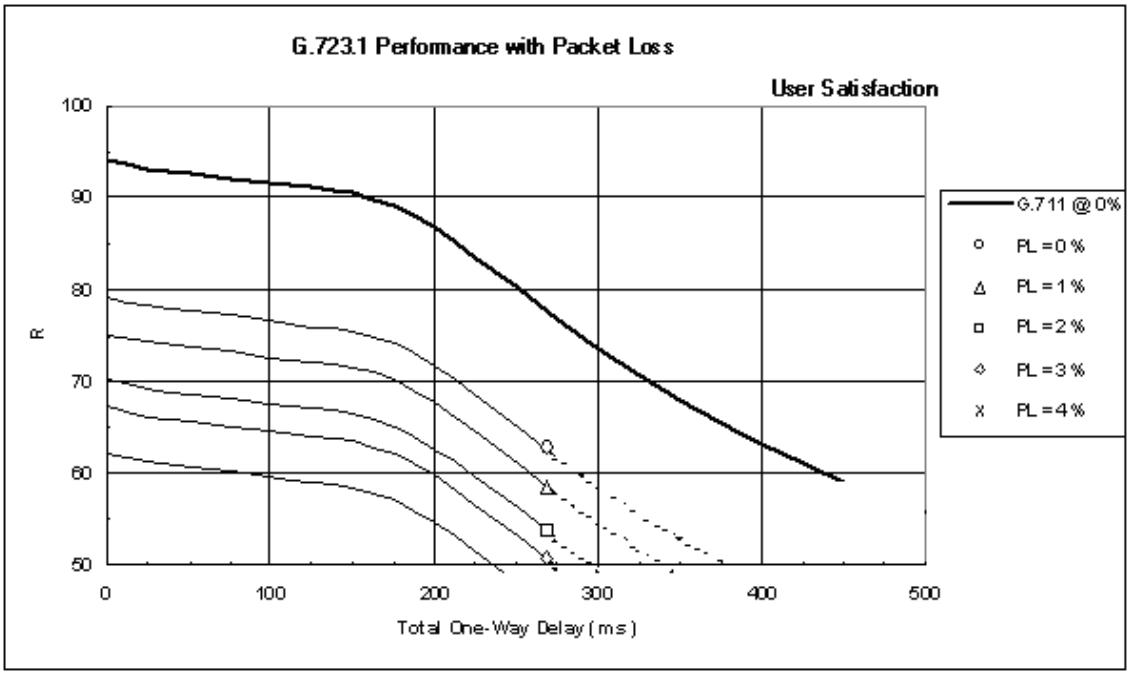


Figure 11: Subjective quality as a function of delay, packet loss and encoding

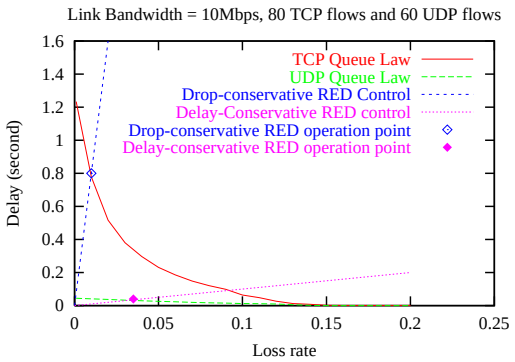


Figure 14: Simple Separation for Static Traffic

queue with a drop-conservative RED configuration as in Figure 4 and UDP traffic is scheduled to a queue with a delay-conservative RED configuration. TCP queue has a WFQ weight of 90% while UDP queue has a weight of 10%, matching the average TCP rate of 9Mb/s and UDP rate of 1Mb/s.

The simulation result is illustrated in Figure 14. As we expected, the operation points are close to the interrection points of queue law curves and RED control curves. TCP traffic achieves a low loss rate of 1% with a delay about 0.8 second. In the other hand, UDP traffic achieves a queuing delay only of 0.04 second with an acceptable loss rate about 3.5%.

The advantage of this scheme compared with the one-queue Best Effort shown in Figure 7 is that Voice/UDP traffic encounters a shorter delay, while the service received by TCP (throughput, delay and loss) is not changed. Although obtaining an advantage without a tradeoff may seem surprising, it can be explained by the fact that the large TCP queue size is an artifact of TCP's congestion control algorithm which can be avoided by the UDP traffic through a simple service separation.

This mechanism is adequate for a network with stable traffic for which the network operator need only adjust the RED parameters and WFQ weights at long intervals (days, weeks). A problem is that usually the traffic in the network is variable in time. It would be unfair for each class of traffic to share the link bandwidth through fixed partitioning. We address this problem in the next section.

# V BEDS ARCHITECTURE FOR DYNAMIC TRAFFIC MIX

In the following, we propose three mechanisms that aim at dynamically balancing the quality of forwarding between the two classes of service. In all these mechanisms, UDP traffic benefits from shorter delay and TCP traffic benefits from lower loss rate.

We study the performance of each mechanism through simulation. In all the simulations, we changed the number of flows both in TCP and UDP traffic to generate a dynamic networking enviroment. The simulation runs for 500 seconds. During the simulation, the number of TCP flows or the number of UDP flows changes according to Table 1.

Table 1: Dynamic TCP and UDP Traffic

| Time | 0s | 100s | 200s | 300s | 400s |
|---|---|---|---|---|---|
| TCP flows | 80 | 80 | 160 | 160 | 80 |
| UDP flows | 60 | 120 | 120 | 60 | 60 |

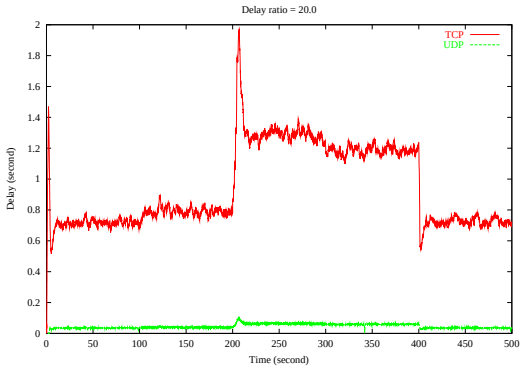


Figure 15: Queuing Delay for Forced Delay Ratio

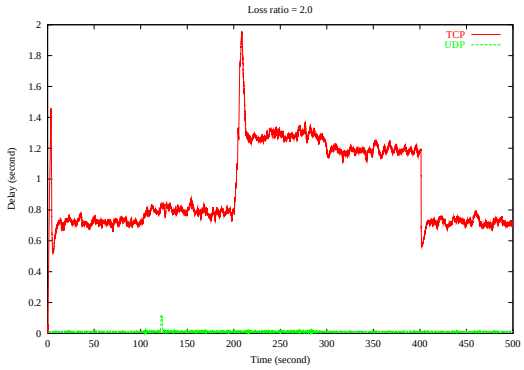


Figure 16: Queuing Delay for Forced Loss Ratio

## A *BEDS scheduling with forced delay ratio*

This mechanism provides service to the queues such that the ratio of their respective queuing delays is close to a preset value for a wide range of traffic intensities. This is achieved with the following algortihm similar to "Backlog-Proportional Rate" proposed in [8].

At each packet arrival, the WFQ weights are adjusted such that

$$\frac{w_{UDP}}{w_{TCP}} = \delta * \frac{q_{UDP}}{q^{+}_{TCP}} \tag{5}$$

where $w_{UDP} + w_{TCP} = 1$ and $q^{+}_{TCP} = \max(q_{TCP}, 1)$.

$q_{TCP}$ and $q_{UDP}$ are the queue length of TCP and UDP queues. $w_{TCP}$ and $w_{UDP}$ are the WFQ weights for TCP and UDP traffic. $\delta$ is a preset delay ratio between TCP traffic and UDP traffic.

For TCP traffic, we use a drop-conservative RED control, and for UDP traffic, we use a delay-conservative RED control. The delay ratio between TCP traffic and UDP traffic is set to 20. Figure 15 illustrates the simulation result. UDP traffic gets 1/20 of TCP queuing delay independent of the traffic changes. The loss rate of the UDP traffic depends on the RED configuration of UDP queue.

The advantage for this mechanism is that UDP traffic will get a much shorter delay than TCP traffic when the delay ratio is set to more than 1. For this mechanism, the problem is that the delay of the UDP traffic and the delay of the TCP traffic will rely on each other. It is unfair that the delay and loss rate of TCP traffic is still high if there is only little TCP traffic and large amount UDP traffic in the network. Another problem is that the loss rate of UDP traffic is not related to the loss rate of TCP traffic. It would be unfair for UDP traffic to get both lower loss rate and shorter queuing delay than TCP traffic.

## B *BEDS scheduling with forced loss ratio*

In this mechanism, the UDP queue is serviced with a higher priority than TCP queue. The TCP queue is also managed by a RED module. Packets entering the UDP queue are dropped with a probability that is multiple (with a preset factor) of the drop probability computed by the RED module at the TCP queue. If the factor is at least equal to one, it is easy to see that this mechanism does not service UDP at a higher rate than in the case of one-queue Best Effort, despite the higher priority of UDP queue.

More precisely, the algorithm is as follows.

1. When a packet enters the packet queue,
   - If it is a TCP packet, the drop rate is calculated from RED control function in TCP queue;
   - If it is a UDP packet, the drop rate is calculated from current TCP drop rate and $\lambda$ (the loss rate ratio), $p_{UDP} = \lambda * p_{TCP}$. $p_{TCP}$ is the drop rate of TCP traffic and $p_{UDP}$ is the drop rate of UDP traffic.
2. Packet scheduling is strict priority, with UDP queue having higher priority.

For TCP traffic, we use the drop-conservative RED control as in Figure 4. For UDP traffic, the loss ratio between UDP traffic and TCP traffic is set to 2.0. Figure 16 illustrates the simulation result. UDP traffic gets an almost zero queuing delay no matter how the traffic changes, since UDP traffic is scheduled with higher priority. TCP traffic gets a delay about 0.8 second and a low loss rate at about 1% when TCP traffic is light, and get a delay of 1.4 second and a loss rate of 1.5% when TCP traffic is heavy. When UDP traffic changes, the operating point of TCP traffic changes only slightly, since the total offered UDP only takes a small fraction of the bandwdith. UDP traffic experiences 2% and 4% loss rate during the corresponding timerespectively for the two traffic conditions.

Compared with the original one queue Best-Effort mechanism in Figure 7, the advantage for "forced loss ratio" is that UDP traffic gets almost zero queuing delay in the router. The operator can change the loss rate ratio to balance the tradeoff between the queuing delay and loss rate of UDP traffic. A problem with this mechanism comes from the very small delay at the UDP queue. If some TCP traffic is sent

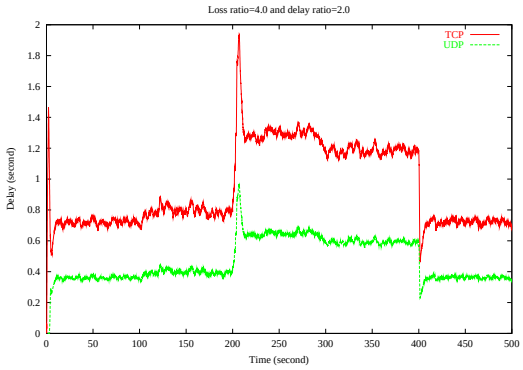


Figure 17: Queuing Delay for Forced Loss Ratio and Delay Ratio

to be serviced through the UDP queue, it receives a much higher service rate than it would otherwise get through the TCP queue, which would be unfair to the TCP flows using the TCP queue. We address this problem in the following.

### C *BEDS scheduling with forced delay and loss ratio*

This mechanism is a combination of the first two with the objective to ensure that both the loss ratio and delay ratio are constant and independent of the intensities of TCP and UDP traffic.

The algorithm is as following. Upon a packet arrival,

- Adjust the WFQ weights so that
$$\frac{w_{UDP}}{w_{TCP}} = \delta * \frac{q_{UDP}}{q_{TCP}^{+}} \quad (6)$$
same as in (5).
- If it is a TCP packet, the drop rate is calculated from RED control function in TCP queue;
- If it is a UDP packet, the drop rate is calculated from current TCP drop rate and $\lambda$ (the loss rate ratio), $p_{UDP} = \lambda * p_{TCP}$.

In our experiment, for TCP traffic, we use the drop-conservative RED control as Figure 4. The loss ratio between UDP traffic and TCP traffic is set to 4.0 and the delay ratio between TCP traffic and UDP traffic is set to 2.0. Figure 17 illustrates the simulation result. UDP traffic gets four times loss rate and half queuing delay of TCP traffic no matter how the traffic changes. TCP traffic gets similar service as in previous experiments.

This mechanism solves the fairness problem of the previous scheme. A TCP flow marked for the UDP queue does not achieve an advantage in service rate. From the model of average TCP throughput in (8) we have that for small packet loss probability,

$$T(p,R) = \frac{M}{R}\sqrt{\frac{3}{2bp}} + o(\frac{1}{\sqrt{p}}) \approx C * \frac{1}{R * \sqrt{p}}. \quad (7)$$

If the round trip time $R$ of a TCP flow is approximately equal to the queuing delay (propagation delay $R_0$ is comparatively small) and if we set the delay ratio $\delta$ and loss ratio $\lambda$ such that $\delta = \sqrt{\lambda}$, then the average throughput for a TCP flow achieved through any of the two queues is approximately the same.

## VI Conclusion

In this paper, we proposed a novel architecture, Best Effort Diferentiate Services, realized with a combination of scheduling and queue management mechanism. We discussed its principles and performed simulations and experiments on a test-bed to prove its benefits. BEDS provides services with two chracteristics: drop-conservative and delay-conservative. Two major classes of traffic get proper differentiated services via scheduling into the two different queues. The WEB/TCP traffic is scheduled into the drop-conservative queue and achieves a lower loss, even though delay may be longer. The VoIP/UDP traffic is scheduled into the delay-conservative queue and achieves a shorter delay, even though loss may be larger. BEDS is a simple mechanism, easy to be implemented in a real router. We demostrated its functionality in a production router.

## VII Acknowledgements

The authors would like to thank Don Towsley of University of Massachusetts, Franco Travostino of Nortel Networks and the members of the "Building Differentiated Services" IRTF group for many useful discussions.

## A Summary of analytic models for TCP and RED

In the following we summarize the model for average TCP throughput $T$ proposed in [29]. $T(p,R) =$

$$\begin{cases} M\frac{\frac{1-p}{p}+\frac{W(p)}{2}+Q(p,W(p))}{R(\frac{b}{2}W(p)+1)+\frac{Q(p,W(p))F(p)T_0}{1-p}} & \text{if } W(p) < W_{max} \\ M\frac{\frac{1-p}{p}+\frac{W_{max}}{2}+Q(p,W_{max})}{R(\frac{b}{8}W_{max}+\frac{1-p}{pW_{max}}+2)+\frac{Q(p,W_{max})F(p)T_0}{1-p}} & \text{otherwise} \end{cases} \quad (8)$$

$T$ is the throughput of a TCP flow (in bits/second) and depends on the packet drop probability $p$, the average round trip time $R$, the average packet size $M$ (in bits), the average number of packets acknowledged by an ACK $b$ (usually 2), the maximum congestion window size advertised by the flow's TCP receiver $W_{max}$ (in packets) and the duration of a basic (non-backed-off) TCP timeout $T_0$ (which is proportional to $R$, typically $5R$). $W$, $Q$ and $F$ have the following expressions:

$$W(p) = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \quad (9)$$

$$Q(p,w) = \min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right) \quad (10)$$

$$F(p) = 1+p+2p^2+4p^3+8p^4+16p^5+32p^6 \quad (11)$$

# References


[1] N. Bhatti, A. Bouch, and A. Kuchinsky. Integrating user-perceived quality into web server design, 2000. HP Technical Report HPL-2003, http://192.6.19.66/techreports/2000/HPL-2000-3.html.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Whang, and W. Weiss. An Architecture for Differentiated Services. RFC 2475, December 1998.

[3] A. Bouch and M. Sasse. Network quality of service; what do users need? In *Proceedings of the 4th International Distributed Conference (IDC'99)*, 1999.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) - Version 1, Functional Specification. Request For Comments (Proposed Standard) RFC 2205, Internet Engineering Task Force, September 1997.

[5] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *IEEE Infocom*, 2000.

[6] K. Claffy, G. Miller, and K. Thompson. The nature of the beast: recent traffic measurements from an Internet backbone. In *Inet98*, 1998.

[7] L. Delgrossi and L. Berger. Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+. Internet RFC1819, August 1995.

[8] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. In *Proceedings of SIGCOMM*, 1999.

[9] B. B. et al. Recommendations on queue management and congestion avoidance in the Interne. IETF RFC 2309, April 1998.

[10] V. Firoiu and M. Borden. A Study of Active Queue Management for Congestion Control. In *IEEE Infocom*, 2000.

[11] V. Firoiu, I. Yeom, and X. Zhang. A Framework for Practical Performance Evaluation and Traffic Engineering in IP Networks. In *IEEE International Conference on Telecommunications*, June 2001.

[12] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4), August 1997.

[13] G. Gallway. Response times to user activities in interactive man/machine computer systems. In *Proc Human Factors Society*, 1981.

[14] R. Guerin and V. Pla. Aggregation and Conformance in Differentiated Service Networks: A Case Study. In *ITC Specialist Seminar on IP Traffic Modeling, Measurement and Management*, 2000.

[15] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. Internet RFC2597, June 1999.

[16] P. Hurley, J.-Y. L. Boudec, and P. Thiran. The Asymmetric Best Effort Service. In *Proceedings of IEEE Globecom*, 1999.

[17] ITU-T. G.723.1 Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3kbit/s, March 1996.

[18] ITU-T. Recommendation G.113. Transmission impairments., 1996.

[19] ITU-T. Appendix I to Recommendation G.113: Provisional planning values for the equipment impairment factor Ie, 1998.

[20] ITU-T. Recommendation G.107. The E-Model, a computational model for use in transmission planning, May 2000.

[21] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. Internet RFC2598, June 1999.

[22] M. Mathis, J. Semske, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communication Review*, 27(3), July 1997.

[23] S. MCanne and S. Flyod. ns-LBL Network Simulator, 1997. Obtain via http://www-nrg.ee.lbnl.gov/ns/.

[24] Microsoft. NetMeeting, http://www.microsoft.com/windows/netmeeting, 1996-1999.

[25] V. Misra, W. Gong, and D. Towsley. A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *Proceedings of SIGCOMM*, 2000.

[26] NortelNetworks. BayStack 350 switch, http://www.nortelnetworks.com/products/02/datasheets/2979.pdf, 1999.

[27] NortelNetworks. BayStack 450 switch, http://www.nortelnetworks.com/products/02/datasheets/3086.pdf, 1999.

[28] NortelNetworks. Contivity 1500 external switch, http://www.nortelnetworks.com/products/02/datasheets/3038.pdf, 1999.

[29] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *IEEE/ACM Transactions on Networking*, 8(2), April 2000.

[30] J. Roberts and S. Oueslati-Boulahia. Quality of service by flow aware networking. In *Philosophical transactions of the Royal Society*, 2000.

[31] S. Shenker, C. Partridge, and R. Guérin. Specification of guaranteed quality of service. Request For Comments (Proposed Standard) RFC 2212, Internet Engineering Task Force, September 1997.

[32] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC2001, Jan 1997.

[33] A. K. Werner Almesberger, Jamal Hadi Salim. Differentiated Services on Linux. Work in progress, draft-almesberger-wajhak-diffserv-linux-01.txt, June 1999.

[34] J. Wroclawski. Specification of the controlled-load network element service. Request For Comments (Proposed Standard) RFC 2211, Internet Engineering Task Force, September 1997.