THE ASYMMETRIC BEST-EFFORT SERVICE

Paul Hurley, Jean-Yves Le Boudec, Patrick Thiran
Institute for Computer Communication and Applications (ICA)
Ecole Polytechnique Fédérale de Lausanne (EPFL)

ABSTRACT

We present Asymmetric Best-Effort (ABE), a novel service to provide a "throughput versus delay jitter" service for IP packets. With this service, every best effort packet is marked as either *Green* or *Blue*. *Green* packets, typically sent by real-time applications such as interactive audio, receive more losses during bouts of congestion than *Blue* ones. In return, they receive a smaller bounded queueing delay.

Both *Green* and *Blue* services are best-effort. The incentive to choose one or other is based on the nature of one's traffic and on traffic conditions. If applications are TCP-friendly, an application sending *Blue* packets will receive more throughput but also more delay jitter, than it would if it sent *Green* packets for a given network state and path.

Service provision at each co-operating router can be achieved by Packet Admission Control (PAC) and scheduling. We develop and simulate an initial algorithm that supports this service whose first results show the feasibility of the service.

I. Introduction

In this paper we present a new service, referred to as Asymmetric Best-Effort (ABE). It consists of a "throughput versus delay" service. Inherent in the service definition is the partition of IP packets into either low delay or low loss. Low delay packets, which are called *Green* packets, are given low delay jitter guarantees through the network without reservation. In exchange, these packets receive more losses during bouts of congestion than *Blue* packets, those which desire low loss. Consequently, and assuming "TCP-friendly" behaviour, sources who choose to be *Blue* receive higher throughput than they would if they had chosen to be *Green*. It is important to emphasise that these are both best-effort services, and the incentive to choose one or other is based on the nature of one's traffic with overall benefit for both traffic types.

Congestion results in delaying data packets and dropping some of them in case of lack of resources. The dropped packets are interpreted in the end systems as a negative feedback which results in the reduction in the emission rate of the sender application. Our approach ensures that at any given time the amount of negative feedback is unequally partitioned between the two different types of traffic such that *Green* traffic receives more negative feedback than *Blue* and hence receives less throughput. In exchange, *Green* traffic is given a shorter bounded queueing delay than *Blue* traffic.

These methods do not necessarily rely on a per-flow information processing. Packets treatment is differentiated only according to a generic packet classification method.

ABE is not an alternative to a reservation or priority service such as those provided in the differentiated service context and, as such, does not warrant comparison with them. Rather, its novelty is in the combination of low delay jitter with reduced throughput for delay jitter sensitive traffic (*Green* packets) and the opposite combination for non-sensitive traffic (*Blue* packets).

In Section II we describe how the service applies to hosts and show simulations to illustrate the benefit for applications in using both traffic types.

In Section III we first outline the support required within routers, namely a combination of packet admission control with queueing. We then describe the first implementation of the service which uses a combination of a Packet Admission Control (PAC) and Earliest Deadline First (EDF) scheduling.

In Section IV we provide simulation results to show that the implementation provides the desired service.

II. Service Description: Host Support

A. Service Description

Negative feedback can be either explicit, e.g. Explicit Congestion Notification (ECN) or implicit, e.g. packet loss. In the remainder of this document we consider packet loss as the method of providing negative feedback. Nevertheless, the approach we describe remains valid in the case of systems using some form of ECN and adaptation can be obtained through re-interpretation.

Each packet is either *Green* or *Blue*. *Green* packets would usually be interactive traffic where packet transfer from end to end must be short and delay variation low. Examples of *Green* traffic include Internet Telephony and videoconferencing traffic, where if the data does not reach the receiving application within a certain time it may well be too late to be useful to it.

Blue packets are typically non-interactive traffic whose end to end delay can be variable and the goal is minimisation of overall transfer time. Examples of

Blue traffic include data traffic (e.g. TCP traffic) and delay adaptive stream-like applications (playback audio and video applications).

We do not specify how the *Green* and *Blue* distinction should be made and leave this as open to definition.

The amount of negative feedback (e.g. packet losses) received by *Green* traffic is greater than that received by *Blue* traffic. The admitted *Green* packets are given a shorter queueing delay.

With this definition, the network-level quality of service (packet loss and delay jitter) received by one of the traffic types cannot be classified as being better than the other. Each traffic type receives a different quality. The appropriate matching between the QoS received and the application nature is a major advantage of this system. This scheme also avoids making an unsatisfactory trade-off between the different buffer size requirements of real and non-real time traffic.

No rate reservation is needed. During a silence period of a given traffic type, the other type can make use of the whole bandwidth. There is benefit for all best-effort traffic types. Delay jitter sensitive traffic, such as voice, receives a low bounded delay, possibly at the expense of reduced throughput; over moderately loaded network paths, this results in superior voice quality; if the path becomes highly loaded, then such a source can always revert to sending blue packets only. Non delay-sensitive traffic, such as image or text transfer, receives a low number of losses. If a source which was previously *Blue* now decides to mark its packets as *Green*, it will receive less delay jitter, but this will never be at the expense of other sources, which continue to send their packets as *Blue*. Thus, unlike differentiated services, traffic management and charging practices remain essentially the same as for a single class, best-effort network.

We assume that *Green* and *Blue* sources are "TCP-friendly" conformant [5], i.e. they do not send more than a TCP source would for the same conditions of loss. The enforcement of friendliness [8], [7] is currently a research problem. ABE neither makes worse nor improves the enforcement problem.

One could envisage colour mixing strategies, where sources send some of their packets as *Green* and some as *Blue*. This would typically be performed at the application level as expected by Application Layer Framing (ALF). We focus for the rest of this paper on the simpler case where a traffic source chooses to be either *Green* or *Blue*.

B. Service Illustration: The Host Point of View

We now illustrate how the service would work from the host point of view. We show that an application that requires low delay jitter does receive it but at the expense of lower throughput. Conversely, an application that does not care about jitter receives overall reduced end to end transfer delay.

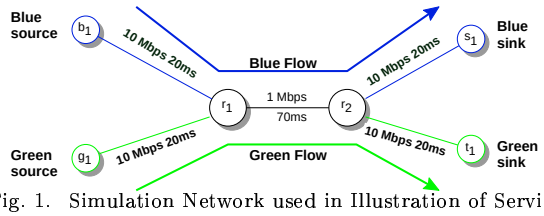

Fig. 1. Simulation Network used in Illustration of Service

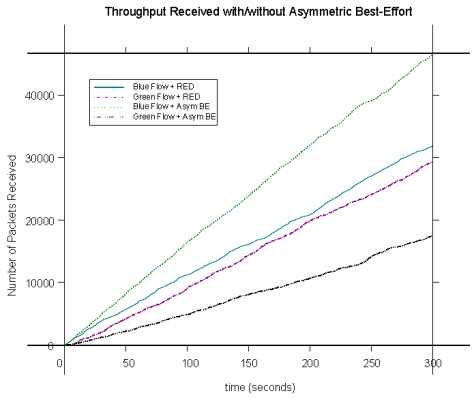

Fig. 2. Number of packets successfully transferred by a *Green* and a *Blue* Source as a function of time. For cases with and without ABE.

The network used in this simulation is shown in Figure 1. A *Blue* and *Green* connection share a bottleneck and the same nonqueueing delay. Router r_1 facilitates the service by the implementation detailed in Section III-B. It uses a modified RED dropping algorithm which is biased towards dropping more *Green* than *Blue* packets and schedules using EDFS such that *Green* traffic receives less waiting time in the queue. A strict bound on *Green* queueing delays is also enforced. The *Blue* source is a TCP Reno source and the *Green* source uses a transport protocol designed to represent TCP friendliness as described in [6].

The EDF scheduler gives priority to *Green* packets by serving packets by smallest tag value first and assigning a tag to *Blue* packets which is D larger than *Green* packets. This value is set to 0.2 for this simulation. The desired throughput distribution ratio, the ratio of throughput between a *Blue* and *Green* source operating under the same conditions, was set to 3.

Figure 2 shows the respective throughputs reached by each source as a function of time in the cases when ABE is used and when a single best effort class is. It is clear from this that the throughput given to the *Blue* source is higher in the ABE case. When not using ABE at time 300 the ratio of *Blue* to *Green* throughput is 1.08. When using it, the ratio is 2.674.

Figure 3 shows the distribution of queueing delays experienced by the *Green* flow at router r_1, again in the case of using ABE and not. We can see the overall

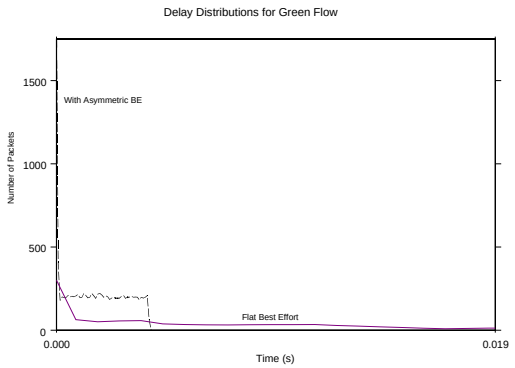

Fig. 3. Queueing Delay for *Green* packets with/without ABE for simple case of one *Green* and one *Blue* flow. General behaviour for many flows is shown in Figure 7.

delay for *Green* is low and varies little when we use the service. The average queueing delay seen for the *Green* flow by using and not using ABE is given by 0.001599 and 0.007449 respectively. Both delays are small which is expected given only two flows, but the ABE average is some orders of magnitude smaller. Also, the standard deviation of the delay for using and not using ABE is 0.001299 and 0.008 respectively, illustrating the increased predictability in the delay received by the *Green* flow.

Overall benefit for each source is achieved. We have shown lower jitter for real-time traffic and higher throughput for file transfer oriented applications. It is good to use *Blue* when one's overall goal is increased average throughput. On the other hand, when one has a real-time constraint, it can be better to choose *Green*.

III. Router Support

A. General Router Requirements

One can consider the traffic control within an IP router in order to provide ABE as being composed of two main algorithms: Packet Admission Control (PAC) and scheduling. The PAC manages the queue by dropping packets whenever necessary or appropriate, acceptance being biased in favour of *Blue* packets. The scheduler determines which packet, if any, from the buffer should be sent next, with bias towards giving *Green* packets a lower delay.

A classifier is also required for identifying the traffic class of the incoming packet i.e. whether it is *Green* or *Blue*.

B. Implementation

A particular version of ABE was designed, and then implemented and simulated in ns [14].

The router mechanism for treatment of packets is outlined in Figure 4. The queue management algorithm uses Random Early Detection (RED) [12] to obtain an initial dropping probability p. RED is a congestion avoidance mechanism, such that when the average queue size exceeds a pre-set threshold, the router drops each arriving packet with a certain probability which is a function of the average queue size.

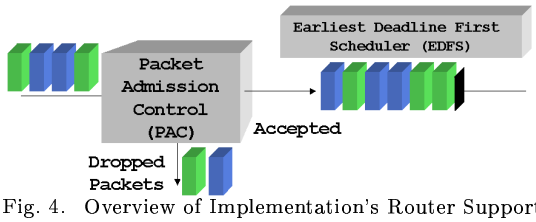

Fig. 4. Overview of Implementation's Router Support

The ideal is considered to be when the distribution of throughput is such that a *Blue* flow receives β^* times as much throughput as a *Green* flow that shares the same path, β^* being an input parameter to the system.

The modification is as follows. The RED dropping probability p is calculated as before. If the packet is *Green*, the probability of dropping is $p*\alpha$. The value of α used is not constant and also depends on the desired throughput ratio β^* of *Blue* to *Green* flows. How α is controlled is explained in Section III-C.1.

RED actually calculates the drop probability in two stages (the second calculated from an input of the first probability and the amount of packets since the last drop) in order to achieve a more uniform distribution of packet losses. The increase in probability of loss for *Green* packets is calculated after the second stage.

The scheduling is Earliest Deadline First [13]. Each packet is assigned a finishing service time deadline, a tag, and the packet currently having the lowest value is served first (i.e. earliest deadline).

Each *Green* packet arriving is assigned a finishing service time deadline equal to the arrival time t. A *Blue* packet is assigned a time equal to the arrival time plus a constant D, namely $t + D$.

This scheduling is more advantageous than a plain priority scheme in which *Green* packets would always be served before *Blue* ones. This is because, by an appropriate setting of D, service starvation for *Blue* traffic can be prevented. The value of D is set such that it reflects the maximum reasonable time a *Blue* packet can spend in the system.

A *Green* packet is provisionally accepted at this point. It then undergoes a second dropping acceptance decision, called *Green* dropping control. It is accepted only if the number of packets in the queue with a deadline less than the current time, referred to as *now*, is less than some defined value L_0. This bounds the queueing delay for *Green* packets to L_0/c where c is the capacity of the output link. In this way, if there are only *Green* packets arriving, the system reverts to a flat best-effort network with smaller buffers. The queueing delay for *Blue* packets is at most $L_{tot}/c + D$ for a buffer size of

L_{tot}.

We summarise the PAC algorithm as follows:

```
For each packet arrival to output port:
  if buffer full
    drop packet
  else if Blue
    drop with random probability p
    if not dropped
      deadline = now + D
      accept packet
  else Green
    drop with random probability alpha*p
    if not dropped
      if N > L0
        drop packet
      else
        deadline = now
        accept packet
```

C. Dropping Control

C.1 Achieving a Throughput Ratio Of β^*

Consider that we would like to offer a service such that, assuming TCP friendliness, the long-term rate of a *Blue* source x_g is $\beta^* \geq 1$ times that of x_b, the long-term rate of a *Green* source who shares the same path i.e.

$$x_b = \beta^* x_g. \tag{1}$$

The question being asked in this section is how can we drop in order to approximately achieve this service goal. It turns out that we can by controlling the ratio in which we drop the packets from the respective classes. We do this by derivation from modelling results which show the relationship between the long term throughput ratio of *Green* to *Blue* traffic which share the same path, and the ratio of packet losses experienced. Since throughput has been shown to be inversely proportional to round-trip time, intuitively we need to compensate for the delay boost *Green* packets experience in order to attain the throughput goal.

There are many well-established formulas which relate long TCP throughput and packet loss ratio [2], [3], [1], [4]. We use a formula, which agrees with but is based on less modelling assumptions than these. It has the advantage of providing an explicit relationship for any magnitude of loss ratio i.e. it varies based on whether the loss ratio is very small or very large. However, our methodology is not dependent on the assumption of an explicit heuristic. Any particular one may be chosen, for either further accuracy, or indeed to reduce complexity.

The following expression [11] shows a relationship between long term throughput for source x_i and packet loss ratio q_i for given modelling assumptions,

$$x_i = \frac{-q_i + \sqrt{2q_i + q_i^2}}{\tau q_i} \tag{2}$$

where τ is the round trip time.

Here we consider calculations and measurements as being in packets per second, thus favouring sending large packets. However, *Green* packets would typically be smaller than *Blue* ones and should one wish to bias more in favour of small packets, one can easily extend to a bytes oriented method.

Note that if the loss ratio is very small this can be approximated by,

$$x_i \equiv_{q_i \to 0} \frac{C}{\tau\sqrt{q_i}}. \tag{3}$$

where $C = \sqrt{2}$. This is the well-known formula of throughput as a function of loss ratio. The only difference is the different constant which resulted from different modelling. The actual value of this constant does not matter in our context as we consider ratios of throughputs.

Define the respective long-term rates of a *Green* and *Blue* source by x_g and x_b respectively. Let the round-trip times for the *Green* and *Blue* source be given by τ and $\tau + B$ respectively. B reflects the fact that a *Blue* source will have a longer round-trip time due to the preferential scheduling for *Green* packets. We refer to B as the (delay) boost *Green* packets receive.

Consider a dropper that is designed such that *Green* sources who share the same path as the *Blue* sources, and receive, on average, α times as many losses as *Blue*, where $\alpha \geq 1$ i.e. the PAC is designed such that the *Green* packet loss ratio, q_g and the *Blue* packet loss ratio, q_b are related by,

$$q_g = \alpha q_b. \tag{4}$$

If q_g and q_b are very small, from Equation (3) we can see that β^* can be approximated by $\sqrt{\alpha}$. When q_g and q_b become large (because there are many flows at the bottleneck), β^* tends towards α. For example, if $\alpha = 2$, on a severely congested path, a *Green* source achieves approximately half the throughput of a *Blue* source. When there is low load, a *Green* source gets approximately 0.707 the throughput of a *Blue* source.

Using Equations (1), (2), and (4) we derive α as a function of β^*, the *Blue* packet loss ratio q_b, the delay boost B and the round-trip time of the *Green* source τ:

$$\alpha = \frac{(\beta^*(1 + B/\tau))^2}{1 + (1 - \beta^*(1 + B/\tau))(q_b - \sqrt{q_b(q_b + 2)})}. \tag{5}$$

This is decreasing in $q_b \in [0, 1]$, with $\alpha = (\beta^*(1 + B/\tau))^2$ when $q_b = 0$. Note that this is a function of τ, the round-trip time for the *Green* source which is neither knowable nor the same for all *Green* sources. This means that, even in the event of the loss-throughput

formula capturing the throughput distribution completely accurately, it remains impossible to achieve exactly a dropping ratio β^* without explicit per-flow information. Instead a system parameter τ_e is chosen to represent the "typical" round-trip time for a *Green* flow. A high τ_e is the more favourable to *Green* flows since it generates a lower α. However, too high a τ_e reduces the throughput given to *Blue* flows.

C.2 The Control Loop

For a given α, a system will not drop exactly α times more *Green* than *Blue* packets due to randomness in the dropping algorithm, variations in the input process and drops due to *Green* dropping control. So in order to approximate a throughput ratio β^*, we must control the value of α. The derivation of such a control law is given in Appendix A. The law is as follows. Let

$$\delta = 1 + (1 - \beta^*(1 + \frac{B}{\tau_e}))(q_b - \sqrt{q_b(q_b + 2)}). \qquad (6)$$

α is updated at each time interval T_1, according to,

$$\alpha \leftarrow \alpha \left(\beta^{*2}(1 + \frac{B}{\tau_e})^2 \frac{q_b}{q_g}\right)^{K_1} \delta^{-K_2}. \qquad (7)$$

If sufficiently many *Greens* have not been dropped we increase α in the next time period, and vice-versa if dropped too many. K_1 and K_2 are chosen sufficiently largely to have a sufficiently strong reaction, while not too large to avoid oscillations.

q_b^m, q_g^m, and B^m are the measurements, in this time period T_1, of the *Blue* loss ratio, *Green* loss ratio, and boost respectively. In order not to be too reactive to the instantaneous values of q_b^m, q_g^m we smooth using an exponentially weighted moving average (EWMA) filter with parameter γ_1.

B^m is also smoothed by a EMWA filter by parameter γ_2. B^m is filtered differently to q_b and q_g in order to react to higher frequency oscillations in this signal.

The initial values for the variables of the system are chosen as follows. q_b is some initial *Blue* loss ratio based on observation (e.g. $q_b = 0.01$). The actual choice, as long as it is non-zero is not that important as the system settles quite quickly. q_g is calculated from the initial q_b. α is calculated from the initial values of q_b and q_g. B is some initial delay boost based on observation. Again, its choice is not crucial for smooth operation e.g. one could simply choose $B = D/2$ where D is the EDFS scheduling parameter. The aspect of parameter choice is touched on in Section IV.

The action performed every T_1 seconds is as follows:

$$\begin{aligned} q_b &\leftarrow \gamma_1 q_b + (1 - \gamma_1) q_b^m \\ q_g &\leftarrow \gamma_1 q_g + (1 - \gamma_1) q_g^m \\ B &\leftarrow \gamma_2 B + (1 - \gamma_2) B^m \end{aligned}$$

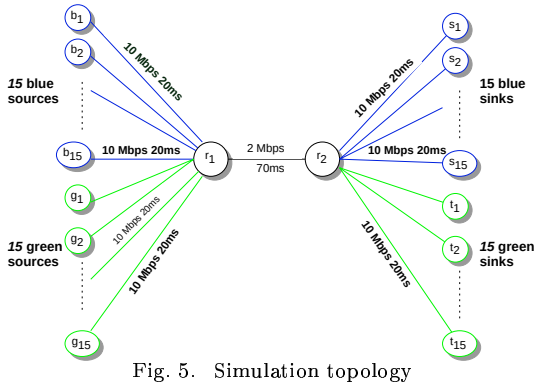

Fig. 5. Simulation topology

$$\alpha \leftarrow \alpha \left(\frac{\beta^{*2}(1 + \frac{B}{\tau_e})^2 q_b}{q_g}\right)^{K_1} \delta^{-K_2}$$

The aspect of parameter choice of parameters is touched on in Section IV.

IV. Simulations

The simulations' goal is to show that the service works in that *Blue* and *Green* traffic are "happier" than they would have been in a flat best-effort service. That is *Blue* traffic receives more throughput then it would from the flat best-effort case, while *Green* traffic receives a low bounded delay while still receiving acceptable throughput.

The TCP friendly protocol is basic, and its success in provided said property is not under scrutiny. Other nongoals of this work, left for future investigation, include the determination of "good" RED parameter sets to support the service and how to choose appropriate combinations of D and β^*. The effect of the control loop parameters is initially looked at, in depth investigation being left for future study.

A. Summary of Simulation Results

Under the conditions we tested, it was shown that *Blue* traffic and *Green* traffic were happier under ABE than under flat conditions.

The implementation provides differential dropping successfully, but is highly sensitive to the choice of RED parameters. If they are wrongly set it can result in a high number of forced losses (not dropped with a probability of less than one) and the throughput differential cannot be controlled very well.

The anticipated long term throughput distribution has been approximately achieved. However, the choice of τ_e determines the level of success in achieving this.

The control loop, under the simple conditions tested, was not very sensitive to the choice of control loop parameters.

B. Simulations Description

The test topology, as shown in Figure 5, consists of 15 *Green* sources and 15 *Blue* sources. Each source shares the same bottleneck and has the same propagation delay to their respective sinks.

One bit in the header of a packet specifies whether the packet is *Green* or *Blue*. The *Blue* source is a TCP Reno source which has always a packet to send. The *Green* source uses the rate-based TCP friendly type algorithm described in [6]. It was a simple one for simulation purposes, used to represent the fact that *Green* traffic will most likely be real-time in nature. More sophisticated rate-based TCP friendly unicast protocols which explore their algorithms ability to provide TCP friendliness effectively are described and evaluated in [10] and [9].

The target throughput distribution ratio β^* of *Blue* to *Green* was 1.5. The *Green* delay bound L_0 was 10 packets. The delay D in the EDFS tagging was set to 0.1 and 0.2 seconds. The round-trip time parameter was $0.14s$, which corresponds to the propogation delay of all sources. The control parameters were $K_1 = K_2 = 1.1$, $T_1 = 0.5$, $\gamma_1 = 0.3$, and $\gamma_2 = 0.5$. The initial values for q_b and B were 0.05 and $D/2$ respectively.

The buffer size at r_1 was 50. The set of RED parameters chosen were $min_{th} = 0$, $max_{th} = 40$, $max_p = 0.2$, and $w = 0.02$. Our simulations with more typical RED parameter settings resulted in many forced losses and hence less control over the dropping differential. When the average queue size is less than min_{th} or greater than max_{th} the probability of dropping a *Blue* or a *Green* packet is the same (either automatically accepted or dropped) and thus in these ranges we lose control of dropping proportionally.

Given that there is a randomness in dropping, average values were obtained after four simulation runs and confidence interval results obtained. Throughput and delay measurements for the first 30s of simulation are not measured as we allow the control mechanism to warm up to reflect more realistic operation.

C. Simulations' Results

Figure 6 shows the average number of packets received by each *Blue* and *Green* connection at a given time. The average at time t is the average obtained over 4 simulation results for that time t. For clarity the confidence intervals are omitted, and the worst-case interval for 95% confidence seen was 19.31 packets.

One can see the throughput benefit given, on average, to *Blue* traffic. The long-term ratio of average *Blue* traffic throughput to *Green* was 1.4506 for $D = 0.1$ and 1.37325 when $D = 0.2$ but when using flat best-effort the throughput was almost the same (ratio was 1.058).

Figure 7 shows how the delay distribution for *Green* packets varies for given values of D. The range of delay narrow, as expected, for increasing values of D. Even for the smaller D (0.1s) we can see a large decrease in the delay variation when compared to the flat best-effort case. Green dropping control in ABE, with parameter $L_0 = 10$, ensures that *Green* delays are strictly bounded.

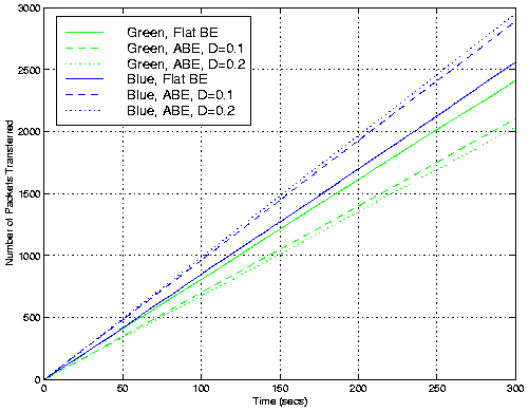

Fig. 6. Average of the average number of packets received per *Green* and *Blue* connection at each time t. For ABE with $D = 0.1, 0.2s$, $\beta^* = 1.5$ and flat best-effort. $K_1 = K_2 = 1.1$, $T_1 = 0.5$

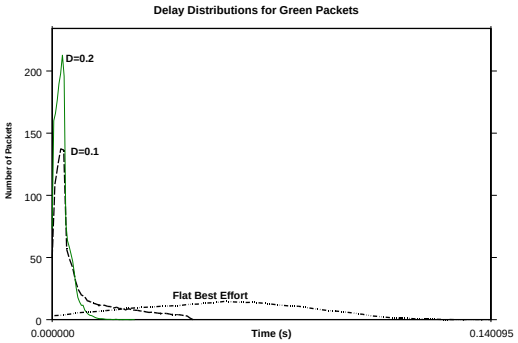

Fig. 7. Delay Distribution received for *Green* packets when $D = 0.1, 0.2$ seconds and flat best-effort.

The throughput results are shown in Figure 8, where all parameters remain the same except $K_1 = 2.0$, $T_1 = 0.1$, and $K_2 = 1.1$. The throughput ratio is again approximately achieved (ratio was 1.4931), showing, in this case, that strictly picked system parameters are not essential to a working ABE. In depth study is required to determine appropriate parameters, based on the distribution of the input traffic.

V. Conclusions and Future Work

We have described a simple but powerful service which enables best-effort traffic to receive requirements closer to its traffic desires yet to everyone's overall benefit. It decouples delay jitter objectives from loss objectives with no concept of reservation or signalling and no change to traffic management or charging. Dimen-

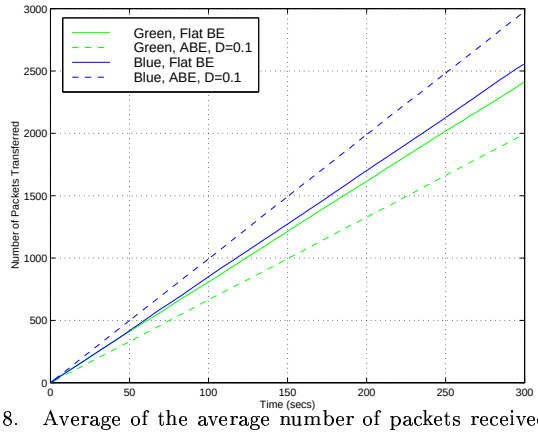

Fig. 8. Average of the average number of packets received per *Green* and *Blue* connection at each time t. Control Parameters are $K_1 = 2.0$, $T_1 = 0.1$, and $K_2 = 1.1$. For ABE with $D = 0.1$, $\beta^* = 1.5$ versus flat best-effort.

sioning the network is also potentially simpler since one would no longer need to choose a buffering compromise to suit both types of traffic.

It should be stressed that ABE is a new service in its own right and not a substitute for reservation or priority services. The service choice of *Green* or *Blue* is self-policing since the user/application will be coaxed into choosing one or the other or indeed a mixture of both, based on its traffic profile objectives.

We presented an initial implementation, which uses a modified version of RED with less probability of dropping *Blue* packets, coupled with an EDF scheduler which favours *Green* packets. It was shown with this implementation that ABE can be achieved, although extensive simulations are needed to completely establish this. In addition, a study is needed to determine optimal control parameter settings for a given input distribution.

The tuning of the parameters of this system, D and β^*, was seen as key to deciding on the respective biases to the traffic types. We are currently designing a system that controls them removing the necessity to specify them. The results were also highly sensitive to the choice of RED parameters. As such, further work is necessary on optimising this system.

Appendix A: Control Law Derivation

We would like to achieve a target throughput ratio of β^*. From Equation (2) and (5), for a given q_b and B the ideal green loss ratio q_g^* is such that,

$$\beta^* = \frac{\tau_e}{\tau_e + B} \frac{q_g^*}{q_b} \frac{q_b - \sqrt{qb(qb+2)}}{q_g^* - \sqrt{q_g^*(q_g^*+2)}}.$$

or equivalently

$$\frac{q_g^*}{q_b} = \frac{(\beta^*(1 + B/\tau_e))^2}{\delta}.$$

where δ is given by Equation (6). One monitors q_g, q_b and B. The goal is to modify α such that $q_g \to q_g^*$ for a given q_b and B. Since $\alpha \approx \frac{q_g}{q_b}$, we can adopt a control law as follows:

$$\frac{\alpha_{t+1}}{\alpha_t} = \left(\frac{q_g^*/q_b}{q_g/q_b}\right)^K = \left(\frac{\beta^{*2}(1+\frac{B}{\tau_e})^2}{\delta}\frac{q_b}{q_g}\right)^K.$$

for some constant $K > 0$, the gain. This can be written:

$$\log \alpha_{t+1} = \log \alpha_t + K\left(\log \beta^{*2}(1+\frac{B}{\tau_e})^2 - \log \frac{q_g}{q_b}\right) - K \log(\delta)$$

To enable fine-tuning we can introduce two separate gains $K_1 > 0$ and $K_2 > 0$, and arrive at the control law:

$$\log \alpha_{t+1} = \log \alpha_t + K_1\left(\log\left(\beta^*(1+\frac{B}{\tau_e})\right)^2 - \log \frac{q_g}{q_b}\right) - K_2 \log(\delta).$$

References

[1] S. Floyd. Connections with multiple congested gateways in packet switched networks, part 1: one way traffic. *ACM Computer Communication Review*, 22(5):30–47, October 1991.

[2] T. V. Lakshman and U. Madhow. The performance of TCP for networks with high bandwidth delay products and random loss. *IEEE/ACM Trans on Networking*, 5(3):336–350, June 1997.

[3] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behaviour of the TCP congestion avoidance algorithm. *Computer Communication Review*, July 1997.

[4] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, Modeling TCP Throughput: A Simple Model and its Empirical Validation. Proceedings of SIGCOMM'98.

[5] TCP friendly web site. `http://www.psc.edu/networking/tcp_friendly.html`

[6] P. Hurley, J. Y. Le Boudec, M. Hamdi, L. Blazevic, P. Thiran. The Asymmetric Best-Effort Service Technical report No. SSC/1999/003, January 1999.

[7] Floyd, S., and Fall, K. Promoting the Use of End-to-End Congestion Control in the Internet. Under submission, February 1998.

[8] B. Suter, T.V. Lakshman, D. Stiliadis, A. Choudhury. Design Considerations for Supporting TCP with Per-flow Queueing. *Proceedings of IEEE INFOCOM'98.*

[9] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. *IEEE INFOCOMM 99.*

[10] Jitendra Padhye, Jim Kurose, Don Towsley and Rajeev Koodli. A TCP-Friendly Rate Adjustment Protocol for Continuous Media Flows over Best Effort Networks. *UMass-CMPSCI Technical Report TR 98-04*, October 1998.

[11] Paul Hurley, Jean-Yves Le Boudec, Patrick Thiran. A Note On the Fairness of Additive Increase and Multiplicative Decrease. *International Teletraffic Congress (ITC-16)*, June 1999.

[12] Floyd, S., and Jacobson, V. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, V.1 N.4, August 1993, p.397-413.

[13] R. Guerin and V. Peris. Quality-of-Service in Packet Networks - Basic Mechanisms and Directions. *Computer Networks and ISDN Systems. Special issue on multimedia communications over packet based networks*, 1998.

[14] ns v2 simulator. See `http://www-mash.cs.berkeley.edu/ns/`