The Asymmetric Best-Effort Service

Paul Hurley [*]
Jean-Yves Le Boudec
Maher Hamdi
Ljubica Blazevic
Patrick Thiran
Institute for Computer Communication and Applications (ICA)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
`http://icawww.epfl.ch`

SSC Technical Report SSC/1999/003

February 1999

Abstract

We present Asymmetric Best-Effort, a novel service to provide a "throughput versus delay jitter" differentiated 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 less delay jitter.

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 achieved by Packet Admission Control (PAC) and scheduling. We develop and simulate an initial algorithm that supports this service. It uses a modified version of RED for packet drop differentiation while scheduling of *Blue* and *Green* packets is facilitated using Earliest Deadline First (EDF). These first results show the feasibility of the service.

Keywords Asymmetric Best-Effort, Active Queue Management, Random Early Detection, Packet Admission Control, differentiated services, real-time traffic, TCP Friendliness.

[*]Contact author: paul.hurley@epfl.ch, Ph. +41-21-693-6626, Fax +41-21-693-6610

1 Introduction

In this paper we present a new service, referred to as Asymmetric Best-Effort. It consists of a "throughput versus delay" differentiated 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 delay jitter 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.

In Section 2 we describe how the service applies to hosts and show simulations to illustrate the benefit for applications using both traffic types.

In Section 3 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 modified Random Early Detection (RED) algorithm and Earliest Deadline First (EDF) scheduling.

In Section 4 we provide simulation results to show that it can provide the service as desired.

2 Service Description: Host Support

2.1 Service Description

Negative feedback can be either explicit, for example Explicit Congestion Notification (ECN) or implicit, for example 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*. It cannot be neither. *Green* packets would usually be interactive traffic where packet transfer from end to end must be short and delay jitter significant. 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 assumed. During a silence period of a given traffic type, the other type can make use of the whole bandwidth.

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 [4], i.e. they do not send more than a TCP source would for the same conditions of loss. The enforcement of friendliness [6, 5] is currently a research problem. Asymmetric Best-Effort 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*.

2.2 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.

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 described in detail in Section 3.2. It uses a modified RED dropping

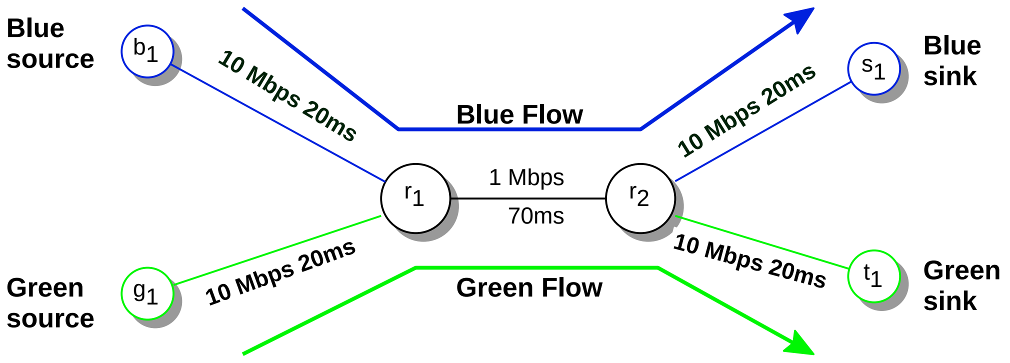

Figure 1: Simulation Network used in Illustration of Service

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.

The *Blue* source is a TCP Reno source and the *Green* source uses a transport protocol designed to represent TCP friendliness which is described in Section 4.1.

The RED parameters chosen were $min_{th} = 0$, $max_{th} = 40$, $max_p = 0.2$ and $w = 0.02$. These RED parameters are ones typically not used, and we discuss this issue again in the more detailed simulations.

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 asymmetric best effort 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 asymmetric best effort case. When not using asymmetric best-effort 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 asymmetric best effort and not. We can see the overall delay for *Green* is low and varies little when we use the service. The average delay seen for the *Green* flow by using and not using asymmetric best-effort is given by 0.001599 and 0.007449 respectively. Both delays are small which is expected given only two flows, but the asymmetric average is some orders of magnitude smaller. Also, the standard deviation of the delay for using and not using asymmetric best-effort 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

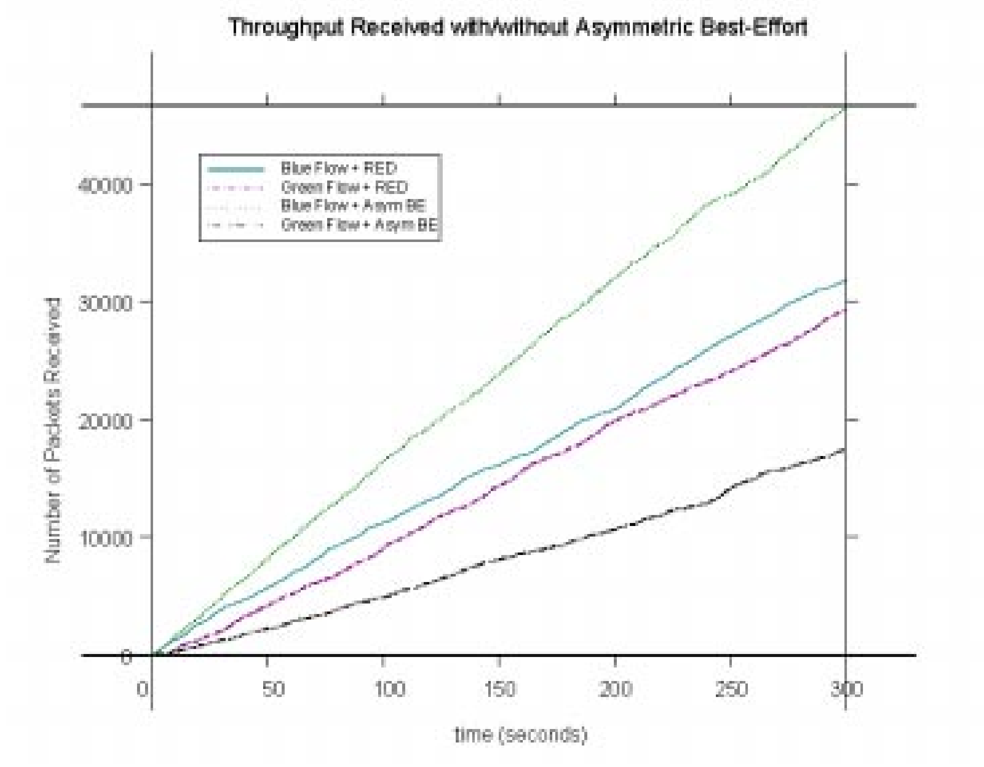

Figure 2: Number of packets successfully transferred by a *Green* and a *Blue* Source as a function of time. For cases with and without asymmetric best effort.

real-time constraint, it can be better to choose *Green*.

3 Router Support

3.1 General Router Requirements

One can consider the support of asymmetric best-effort within an IP router as facilitated by the abstraction into the different modules shown in Figure 4. The traffic control is composed of two main algorithms: Packet Admission Control (PAC) and scheduling. The PAC manages the queue by dropping 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.

The Packet Differentiator is responsible for identifying the traffic class of the incoming packet i.e. whether it is *Green* or *Blue*. The Parameter Collector collects information such as the traffic profile of incoming packets and buffer occupancy. It then provides this feedback,

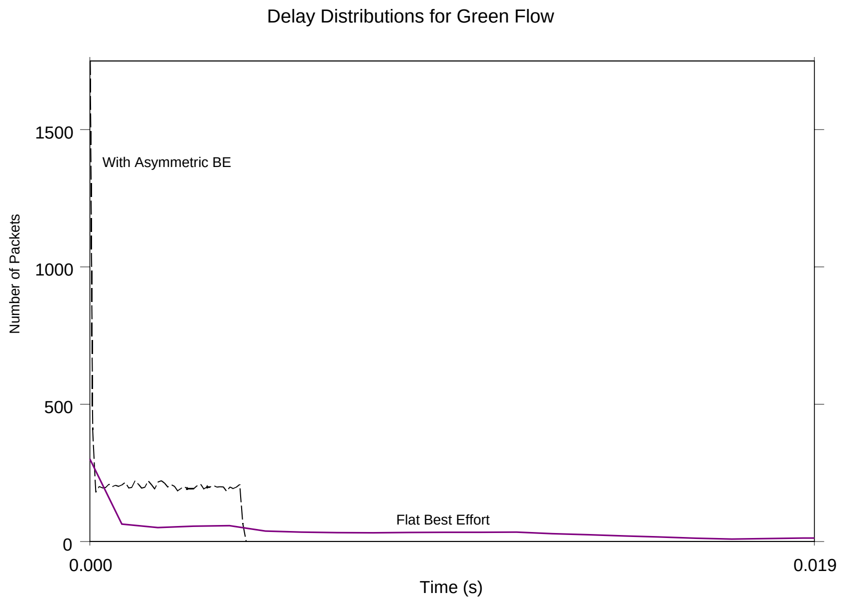

Figure 3: Queueing Delay for *Green* packets with/without asymmetric best-effort for simple case of one *Green* and one *Blue* flow. General behaviour for many flows is shown in Figure 9.

possibly after some processing, to the PAC, so that it may adjust its packet dropping strategy.

3.2 Implementation

A particular version of asymmetric best effort was designed, and then implemented and simulated in ns [12].

The router mechanism for treatment of packets is shown in Figure 5. The queue management algorithm is a modified version of Random Early Detection (RED) [10]. 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.

The service goal is the distribution of throughput such that *Blue* traffic would receive β times as much throughput as *Green* traffic that shares the same path, β being an input parameter to the system.

The modification is as follows. The RED dropping probability is calculated as before. If the packet is *Green*, the probability of dropping is multiplied by α.

The value of α used is not constant and also depends on the desired throughput ratio β of *Blue* to *Green* flows. This is explained in Section 3.3.2.

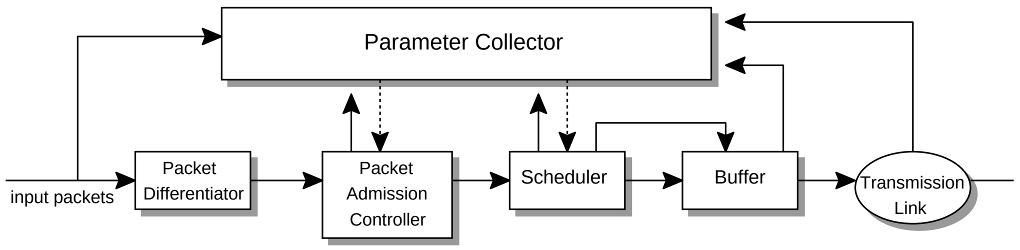

Figure 4: Generic packet processing module within a router

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 [11]. 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.

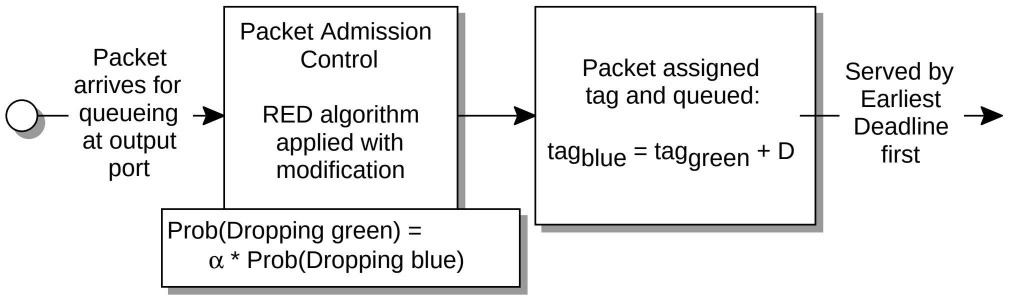
Packet arrives for queueing at output port
Packet Admission Control
RED algorithm applied with modification
Prob(Dropping green) = α * Prob(Dropping blue)
Packet assigned tag and queued:
tag_blue = tag_green + D
Served by Earliest Deadline first

Figure 5: Overview of Implementation's Router Support: For parameters α and D

3.3 Dropping Control

3.3.1 Controlling α: The Ratio Of *Green* to *Blue* Drop Ratios

Suppose the required goal is to have *Green* packets dropped on average α_t times more than *Blue* ones where $\alpha_t \geq 1$. That is,

$$q_g = \alpha_t q_b \tag{1}$$

where q_g is the *Green* packet loss ratio, and q_b the *Blue* packet loss ratio.

In order to do this we must have a controlling mechanism which adjusts the rate of dropping α according to the measured ratio α_m and the target ratio of dropping ratios α_t. The determination of an optimum mechanism is the subject of on-going work. For this implementation we focus on the simple control mechanism where the dropping probability at any instant for *Green* packets is α times as many as *Blue* ones where α is given by,

$$\alpha = \frac{\alpha_t^2}{\alpha_m}. \tag{2}$$

where $\alpha_m = \frac{q_g}{q_b}$ is the ratio of measured green to blue loss ratio.

In this way, we assure that more packets are dropped when our target value is above our measured value and that less are dropped when it is below it. The actual choice of α_t is discussed now in Section 3.3.2.

3.3.2 Achieving a Throughput Ratio Of β

Ideally, it is the throughput ratio not directly the dropping ratio we would like to control. As such, we propose a mechanism for choosing the target dropping ratio α_t such that a desired throughput ratio, for *Green* and *Blue* flows sharing the same path, can be approximately maintained.

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.

$$x_b = \beta x_g. \tag{3}$$

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.

There are many well-established formulas which relate long TCP throughput and packet loss ratio [2, 3, 1]. We use a formula, which agrees with but is based on less modelling assumptions than these. However, it has the advantage of providing an explicit relationship for any of loss ratio i.e. it varies based on whether the loss ratio is very small or very large.

The following expression [9] shows a relationship between long term throughput for source x_i and packet loss ratio q_i for given modelling assumptions,

$$x_i = \frac{-\tau q_i r_0 + \sqrt{4 r_0 \tau q_i \eta + \tau^2 q_i^2 r_0^2}}{2 \tau q_i \eta} \tag{4}$$

where τ is the round trip time, r_0 an additive increase parameter, and η a multiplicative decrease parameter (e.g. for TCP $\eta = 0.5$ and $r_0 = \frac{1}{\tau}$ in packets per second).

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,

$$x_i \equiv_{q_i \to 0} \sqrt{\frac{r_0}{\tau q_i \eta}}. \tag{5}$$

Define the respective long-term rates of a *Green* and *Blue* source as x_g and x_b respectively.

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

$$q_g = \alpha q_b. \tag{6}$$

If q_g and q_b are very small, from Equation (5) 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.

Let $r_0 = \tau$ and $\eta = 0.5$ since these sources are considered TCP-friendly. Using Equations (3), (4), and (6) we can derive α as a function of β and the *Blue* packet loss ratio q_b,

$$\alpha_t(\beta, q_b) = \frac{\beta^2}{1 + (1 - \beta)(q_b - \sqrt{q_b(q_b + 2)})}. \tag{7}$$

This is decreasing in $q_b \in [0, 1]$, with $\alpha = \beta^2$ when $q_b = 0$.

This formula does not provide a general target ratio since q_b varies with time. Instead, to achieve the service goal of a distribution of rates of approximately β, we consider Equation (7), at a given instant, as providing the estimated target value, thus providing a moving target, so to speak.

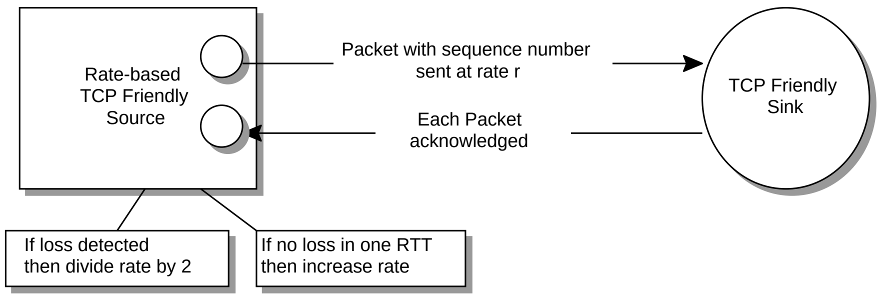

Figure 6: TCP Friendly Protocol

The above analysis assumes that both *Green* and *Blue* traffic sharing the same path would have approximately the same round-trip time. This is not the case, since *Green* packets receive queueing priority, and the simulation results show that the throughput ratio is affected by D. As such, we are currently working on a method such that the ratio of dropping α is also a function of the EDF scheduling parameter D.

4 Simulations

4.1 A TCP-Friendly protocol

Green traffic would typically be rate rather than window based, and not necessarily concerned with loss recovery given its real-time nature. As such, a protocol was designed which approximates TCP friendliness in a rate based unicast context. This is illustrated in Figure 6. The goal was not to define a sophisticated rate based scheme which is proven to be TCP-friendly. Rather it was to have a simple one for simulation purposes. More sophisticated rate-based TCP friendly unicast protocols which explore their algorithms ability to provide TCP friendliness effectively are described and evaluated in [8] and [7].

Each packet contains a sequence number. The receiver acknowledges every packet it receives. A source starts by sending one packet. It is then subjected to additive increase, multiplicative decrease as follows. In the event of a loss, the rate, measured in packets per second, is reduced by half to $r/2$. No retransmission is attempted. If in a given round trip time, τ, no loss is detected, the source increases its rate by $1/\tau$.

The round trip time is estimated by the same algorithm as TCP. Losses are detected by two means. Each packet sent is assigned a timeout and a loss is deduced if one of these timeouts expire before receipt of an acknowledgement for that packet.

In the spirit of the Fast Retransmit algorithm in TCP, losses are also deduced when we receive two acknowledgements and a gap remains in the sequence number acknowledgement space. A loss (or losses) causes a restart of all timers for packets up to the most recently acknowledged, and monitoring is restricted to packets after this value.

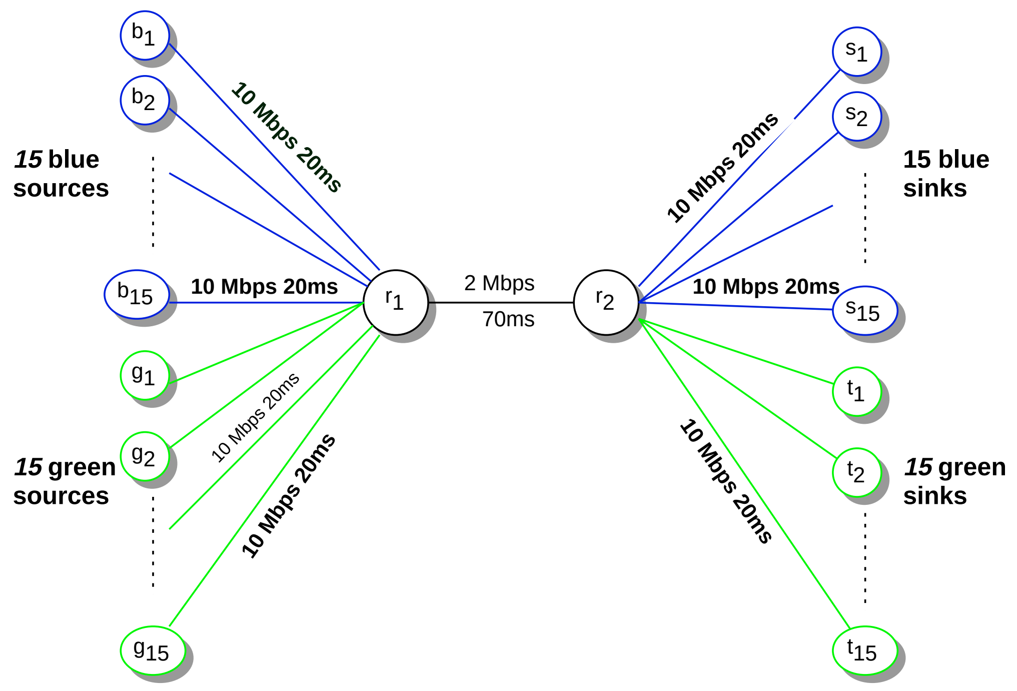

Figure 7: Simulation topology

4.2 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.

We will also show investigate how well the implementation provides differential dropping.

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.

4.3 Simulations Description

The test topology, as shown in Figure 7, 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 TCP friendly type algorithm described in Section 4.1.

A target throughput distribution ratio β of *Blue* to *Green* was set to 3. The delay D in the EDFS tagging was set to $0, 0.1, 0.2, 0.3$ seconds.

The buffer size at r_1 was set to 60. 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 RED is based on randomness in dropping, average values were obtained after four simulation runs and confidence interval results obtained.

4.4 Summary of Simulation Results

Under the conditions we tested, it was shown that *Blue* traffic and *Green* traffic were happier under asymmetric best-effort 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 could be said to have been approximately achieved. However, the ratio accuracy was highest when there was no delay differential, and its accuracy decreased as D increased. This is a result of a knock-on effect of *Green* flows achieving lower delay and thus ending up with higher throughput than otherwise expected.

4.5 Simulations Results

Figure 8 shows the average number of packets received by each *Blue* and *Green* connection at a given time. The average at a given 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 0.823 packets.

One can see clearly the throughput benefit given, on average, to *Blue* traffic. The long-term (taken at time $t = 300s$) ratio of average *Blue* traffic throughput to green was 2.384 but when using flat best-effort the throughput was almost the same (ratio was 1.014).

Figure 9 shows how the delay distribution varies for given values of D. The concentration of delay improves, as expected, for increase values of D. Even when D was small (0.1s) we can see a large decrease in the delay jitter when compared with the flat best-effort case (shown as $D = 0s$).

Figure 10 shows how the average throughput received for different values of D. Here the worst case confidence interval was 0.612 packets for 95% confidence. For all values of D shown the service still provides *Blue* connections with a higher throughput than *Green* .

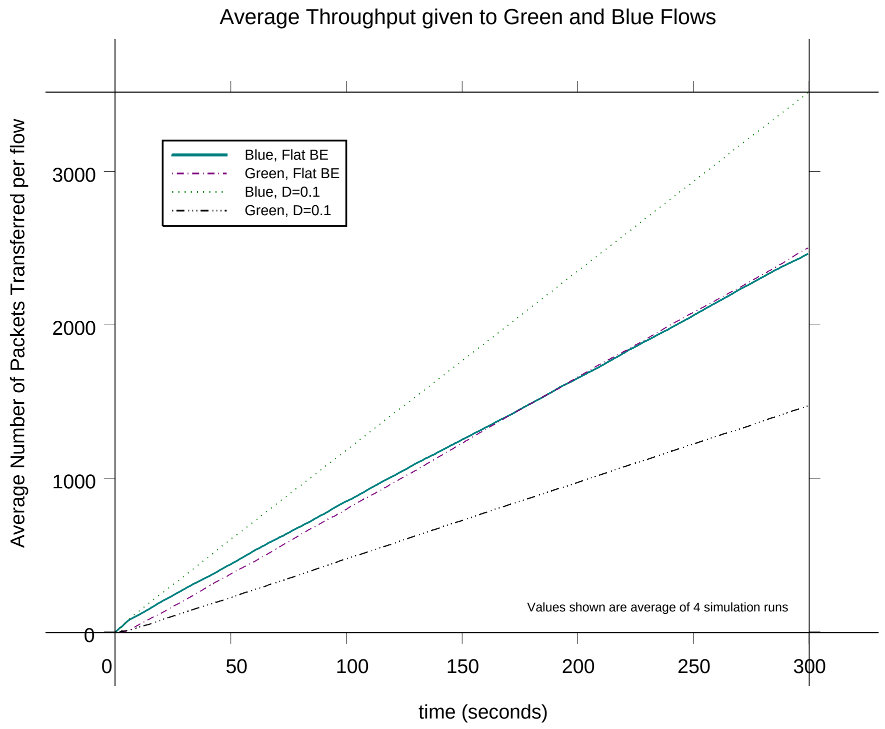

Figure 8: Average of the average number of packets received per *Green* and *Blue* connection at each time t. For Asymmetric Best-Effort with $D = 0.1s$, $\beta = 3$ and flat best-effort.

However, as this throughput differential decreases with D, an optimisation to cope with this needs developing.

5 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. Dimensioning 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 Asymmetric Best-Effort 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.

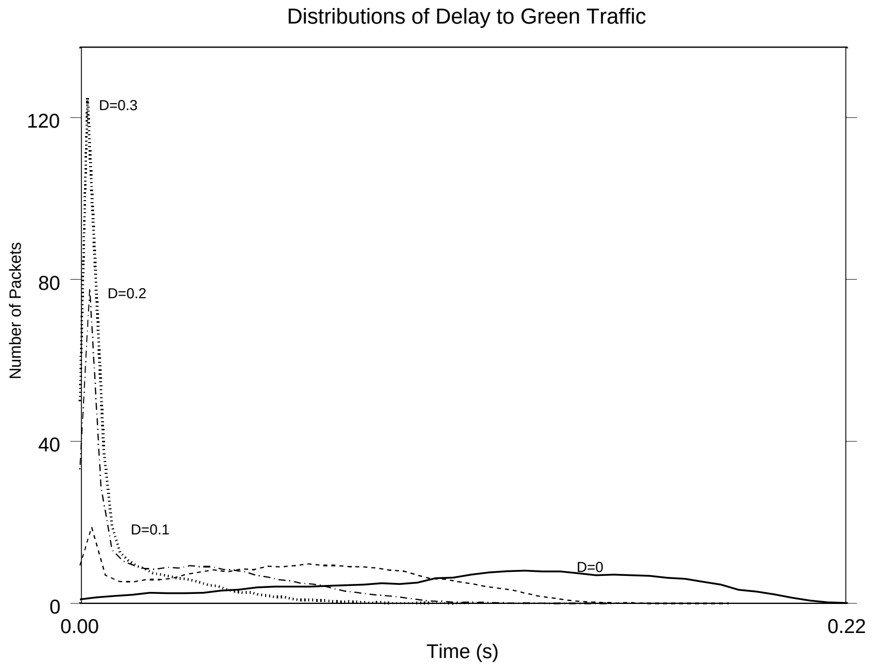

Figure 9: Delay Distribution received for *Green* packets when $D = 0, 0.1, 0.2, 0.3$ seconds (where D=0 signifies flat-best effort).

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 the asymmetric best-effort service can be achieved.

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, and The results were also highly sensitive to the choice of RED parameters. As such, further work is necessary on optimising this system.

In Appendix 1, we present the first step in that direction. Here, the dropping mechanism is based on estimation of the deterministic effective bandwidth and deterministic equivalent bandwidth and whose estimation methods are controlled by one or a set of virtual systems.

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.

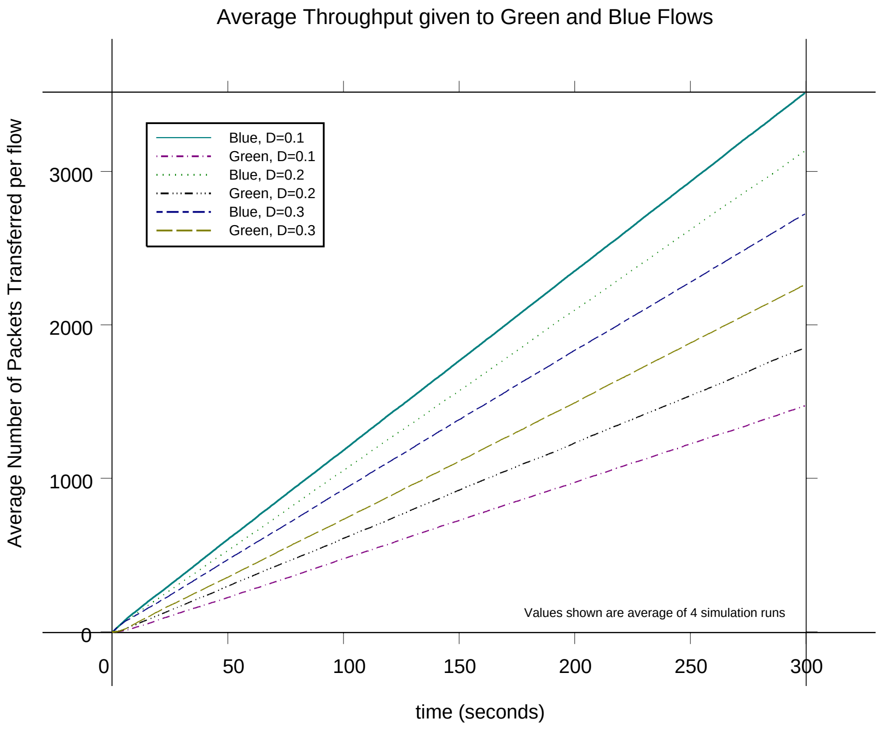

Figure 10: Average of the average number of packets received per connection for each time t for Asymmetric Best-Effort when $\beta = 3$ and $D = 0.1, 0.2, 0.3s$.

[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*, 3, July 1997.

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

[5] Floyd, S., and Fall, K. Promoting the Use of End-to-End Congestion Control in the Internet.

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

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

[8] 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.

[9] 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, To Appear.

[10] 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.

[11] 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.

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

[13] J.Y. Le Boudec. Application of Network Calculus to Guaranteed Service Networks. *IEEE Transactions on Information Theory*, 44(3), May 1998.

Appendix 1 : Framework for a Next Stage Implementation

Here we describe a collection of algorithms for estimation and control that can be used in routers to optimize the asymmetric best effort service.

Let $e_G(t) > 0$ and $e_B(t) > 0$ be estimations of the instantaneous arrival bit rate at time t of *Green* and *Blue* traffic respectively.

Let $C > 0$ be the service bit rate of the output port.

Consider again the target loss objective as specified by Equation (1) in Section 3.3.1.

From this, for any time t, the required probability that a *Blue* and *Green* packet should be dropped are given respectively by

$$\text{Prob(Dropping a } \textit{Blue} \text{ packet)} = (1 - \frac{C}{e_G(t) + e_B(t)}) \frac{e_G(t) + e_B(t)}{e_B(t) + \alpha_t e_G(t)}$$

and

$$\text{Prob(Dropping a } \textit{Green} \text{ packet)} = \alpha_t * \text{Prob(Dropping a } \textit{Blue} \text{ packet)}.$$

Provided we have algorithms to estimate and control $e_G(t)$ and $e_B(t)$, the Packet Admission Controller can meet the target loss objective by using these drop probability functions for acceptance decisions. To this end, we now describe some specific algorithms.

Estimation

Consider the two following estimation procedures.

- This is based on the computation of the deterministic *effective bandwidth* [13]. If $\alpha(s)$ is an arrival curve of the estimated traffic and D a delay bound, the effective bandwidth E^D is defined as,

$$E^D = \sup_{s \geq 0} \frac{\alpha(s)}{s + D}. \tag{8}$$

  In order to approximate an instantaneous estimation, the effective bandwidth must be computed on a sliding window of time w, and is hence given by,

$$E^D(t) = \sup_{t-w \leq t_i \leq t_j \leq t} \frac{P_i + ... + P_j}{t_j - t_i + D}$$

  where P_i is the size of the packet that arrives at time t_i.

- This procedure is based on the computation of the deterministic *equivalent bandwidth* [13]. If $\alpha(s)$ is an arrival curve of the estimated traffic and B a buffer size, the equivalent bandwidth F^B is given by,

$$F^B = \sup_{s \geq 0} \frac{\alpha(s) - B}{s}. \tag{9}$$

  Similarly, the equivalent bandwidth needs to be computed on a sliding window of time w and is given by

$$F^B(t) = \sup_{t-w \leq t_i \leq t_j \leq t} \frac{P_i + ... + P_j - B}{t_j - t_i}$$

  where P_i is the size of the packet that arrives at time t_i.

Broadly speaking, the difference between the two methods is that in the first one, the smoothing effect of the estimation results in a bounded delay D [1] while in the second method it results in a bounded buffer size B [2]. The choice of method depends on whether the constraints are in terms of delay or buffer size.

The estimation of *Green* traffic is performed using the effective bandwidth approach whereas the delay parameter D is fixed according to the target maximum delay. On the other hand, the equivalent bandwidth is used in the estimation of *Blue* traffic. The estimation parameter B then corresponds to the actual buffer size at the output port.

Exponentially Weighted Moving Average (EWMA) is a known low-pass filter that can be used to determine an average value of a measured parameter. Given a series of measured values $\{X_i\}$, the EWMA is given by,

$$Y_i = aY_{i-1} + (1 - a)X_i$$

[1] defined by the maximum delay encountered by the traffic as if it were serviced at a rate of E^D

[2] defined by the maximum buffer size that arises as if the estimated traffic were serviced at a rate of F^B

where a is a parameter to control the smoothing of the filter.

The values of $E^D(t)$ (respectively $F^B(t)$) are averaged using an EWMA filter to give a raw estimation function (for simplicity the same notation $E^D(t)$ and F^B that henceforth refers to the EWMA filtered values is kept).

Accuracy Control

The estimation can temporarily fail to track the instantaneous arrival rate. Accuracy control is needed because the system expressions are entirely based on estimated values. It is based on tracking the state of a virtual system that measures the distance between the actual traffic and the estimated values and react to the latter to give an adjusted estimation function which we denote $e(t)$ (the G and B indexes are removed for now). The *unexpected losses* seen by scheduled packets are used as additional feedback. Let $E(t)$ denote the raw estimation function (either $E^D(t)$ or $F^B(t)$). The adjusted estimation function $e(t)$ is given by,

$$e(t) = E(t)\beta(t)$$

where $\beta(t)$ is the accuracy control factor. $\beta(t)$ is controlled by the state of a virtual system and the amount of unexpected losses $UL(s,t)$.

Consider the following two procedures for accuracy control using virtual systems.

- A virtual buffer is continuously emptied at a rate $e(t)$ and is filled as the packets of the estimated traffic arrive by the equivalent number of bits. Let $X(t)$ denote the virtual buffer fullness at time t. $X(t)$ is updated at each packet arrival as follows:

  $$X(t_i) = P_i + \max(0, X(t_{i-1}) - e(t_{i-1})(t_i - t_{i-1})).$$

  The correction factor is updated periodically, every Δ period of time, using the control:

  $$\beta(t+\Delta) = \beta(t) + K_1 \frac{UL(t, t+\Delta)}{\Delta e(t)} + K_2 \frac{\sup_{t \le s \le t+\Delta} X(s) - B}{e(t)}$$

  where $K_1 > 0$ and $K_2 > 0$ are constants tuned by the implementation.

- Here we use two virtual buffers, with the first one the same as in the first procedure. The second one is continuously filled at a rate $e(t)$ and emptied each packet arrival by the equivalent amount of bits. Let $Y(t)$ denote its fullness at time t. $Y(t)$ is updated each packet arrival,

  $$Y(t_i) = e(t_{i-1})(t_i - t_{i-1}) + \max(0, Y(t_{i-1}) - P_i).$$

  The accuracy control factor is updated upon each packet arrival using the control,

  $$\beta(t_i) = \beta(t_{i-1}) + K_1 \frac{UL(t_{i-1}, t_i)}{(t_i - t_{i-1})e(t)} + K_2 \frac{X(t_i)}{e(t_{i-1})} - K_3 \frac{Y(t_i)}{e(t_{i-1})}$$

  where $K_1 > 0$, $K_2 > 0$ and $K_3 > 0$ are tuned constants.