ABE: Providing a Low-Delay Service within best-effort

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

September 2000

Abstract

We propose Alternative Best-Effort (ABE), a novel service for IP networks, which retains the simplicity of the original Internet single class, best effort service, while providing low delay to interactive, adaptive applications. With ABE, applications choose between either a lower end-to-end delay or more overall throughput. Every best effort packet is marked as either *green* or *blue*. Green packets are guaranteed a low bounded delay in every router. In exchange, green packets are more likely to be dropped (or marked using congestion notification) during periods of congestion than blue packets. For every packet, the choice of one or other colour is done by the application, based on the nature of its traffic and on global traffic conditions. Typically, an interactive application with real-time deadlines, such as audio, will mark most of its packets as green, as long as the network conditions offer a large enough throughput. In contrast, an applications that is transferring binary data, such as bulk data transfer, will seek to minimise overall transfer time and send blue traffic. There is benefit for all traffic in that green traffic achieves a low delay and blue traffic receives at least as much throughput as it would in a flat, that is existing, best-effort network. The key feature of ABE is that neither packet colour can be said to receive better treatment, thus flat rate pricing may be maintained, and there is no need for reservations or profiles. ABE is thus different from differentiated or integrated services: it offers no priorities, reservations or guarantees, but it offers *a new dimension* to best-effort services. In this paper, we define the ABE service, its requirements, properties and usage. We discuss the implications of *replacing* the existing IP best effort service by the ABE service, and analyse in particular the relationship with TCP friendliness. We identify the ABE router requirements. We propose and implement a method for supporting the ABE service at the output port of a router. We discuss its algorithmic aspects and compliance to the ABE router requirements, and present initial simulation results.

Keywords Alternative Best-Effort, ABE, Traffic Control, best-effort, low delay service, real-time traffic, TCP Friendliness.

1 Introduction

We present an enhancement to the IP best-effort service, Alternative Best-Effort (ABE). The goal of ABE is to (1) provide a low queueing delay service and (2) operate in best effort mode, retaining the simplicity of today's best effort Internet service. The first requirement is for applications with stringent real time constraints, such as interactive audio. The second requirement is an attempt to maintain the simplicity of the original Internet. With ABE, it is not required to police how much traffic uses the low delay capability; as explained below, the service is designed to operate equally

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

[†]School of Computing, University of Leeds, United Kingdom

well in all traffic scenarios. The aim of this paper is three-fold:

- To introduce the ABE service, its objectives, requirements and properties;
- to demonstrate its application and usefulness in the context of best-effort services;
- to illustrate the implementation of the ABE service

ABE is designed primarily to support rate-adaptive multimedia applications in a best-effort service environment. These applications adapt to a network state: the rate is reduced when negative feedback is received, and increased with positive feedback. In today's Internet, feedback is based on packet drop. In the future, binary feedback based on Explicit Congestion Notification (ECN) [17] will be used. We assume, as is required in the Internet, that rate adaptation is performed such that the application is *TCP-friendly* [7], namely, it does not receive more throughput than a TCP flow would. It is now established that it is possible to implement adaptive multimedia applications, which perform across a wide range of network conditions [18, 19]. However, delay remains the major impediment for interactive applications in many circumstances [22]. In this context, the key idea of ABE is to provide low-delay at the expense of maybe less throughput. As discussed below, this second point is fundamental in ensuring that ABE requires no usage control.

ABE operates as follows. Best effort IP packets are partitioned into either low delay packets, called *green* packets, and other best effort packets, called *blue* packets. The choice of the terms blue and green, two primary colours of equal value, is to indicate that none of the two has priority over the other, while green, the colour of the traffic light signal for *go*, indicates low queueing delay.

Green packets are given a guarantee of a low bounded delay in every router. In exchange, these packets receive more losses during bouts of congestion than blue packets. If ECN is used, then green packets are more likely to be marked with the congestion bit than blue packets. For simplicity, in the rest of the paper we consider only non ECN-capable systems. Nevertheless, ABE is equally valid, and indeed would work even better with ECN.

It is important to note that ABE addresses a different market to that of differentiated or integrated services. Unlike these services, ABE does not offer any guarantee, even approximate, on throughput but it offers a new degree of freedom to best-effort services. ABE enables a moderately loaded network to offer low delay to some applications (typically, adaptive multimedia applications), as long as such applications are satisfied with the throughput they receive. However, a highly loaded network offering ABE will give little throughput to all best effort flows, no matter whether green or blue.

We describe a scheme, Duplicate Scheduling with Deadlines (DSD), for the implementation of the ABE service at the output port of a router. It enables ABE through the use of deadlines and a virtual queue.

It was implemented in the ns-2 simulator [11]. Its degree of compliance with respect to the ABE service requirements is shown. Based on these initial experiments, the simulation results clearly show the advantage of the ABE service over a flat best-effort service. We discuss end-to-end delay distributions for green traffic and throughput for blue traffic and compare these to a best-effort service. By simulation, we show that an ABE service provides more throughput to blue packets than under a flat best-effort network while giving a low bounded delay to green packets.

The remainder of this paper is organised as follows. Next Section reviews related work and position ABE with respect to other proposals in differentiated services. Section 3 presents the ABE service, its requirements, its properties and a detailed example. It also discusses migration issues from the traditional IP service (flat best-effort) to ABE as well as some marking strategies for applications. In Section 3.4 we discuss the central issue of *green does not hurt blue* and its relation to TCP-friendliness. We discuss router implementations in Section 4, outline an implementation proposal, discuss its compliance to the ABE requirements and show simulation results. In Section 5, we conclude.

2 Other Services for low delay in the Internet

A number of propositions exist for providing low delay in the Internet. A first family of solutions, Integrated Services, uses reservations; this is a considerable departure from the current Internet flat rate philosophy, as it requires per flow accounting and charging.

A second family of solutions, Differentiated Services, uses some form of priority. Crowcroft [1] proposed a low delay service, analysed by May et al [12], coded with a single bit. Turning on this bit ensures that the packet receives serving priority while constrained to a smaller buffer size. Depending on the input traffic and the buffer sizes of both types of traffic, this typically would result in the low delay traffic also having more throughput. Similarly, Expedited Forwarding [14] (EF) aims at providing extremely low loss and low queueing delay guarantees. SIMA [20] offers applications the choice of a level (0-7) of how "real-time" its traffic is, with each level having relatively lower delay and loss ratio than the previous one. Dovrolis et al [16] and Moret and Fdida [13] both describe a system based on a proportional distribution model, where the quality between classes of traffic is proportional and thus can be performed independently of the load within each class. They both propose controlling the relative queueing delays between classes. All of these proposals couple low delay with improved throughput, and are some form of priority. They can be used to support adaptive and non-adaptive interactive applications, provided that some form of admission control is performed. They can provide a premium service, at a price that has to be higher than the best effort service (otherwise all traffic would use the better service). In contrast, ABE green packets cannot be said to receive a better treatment than blue ones and ABE may be introduced as a replacement for the existing best effort service. On the other hand, ABE is not suited to support non-adaptive multimedia applications.

Assured Forwarding (AF) [15] is also a differentiated service. It divides AF traffic into classes within each there are distinct levels of drop precedence. It offers an assurance that IP packets are forwarded with high probability as long as the aggregate input traffic within a class does not exceed an agreed profile. The authors also suggest that an AF class could be used to implement a low delay service where low loss is not an objective, by allocating an AF class with a low buffer space (call it the low delay AF class). Such a service is in principle different from ABE, which views all blue and green packets as one class; the service received by green packets is dependent on the amount of green *and* blue traffic. In contrast, the performance of an AF low delay class is not expected to be affected by the amount of best effort traffic. In that sense, the low delay AF class is a differentiated service which requires differentiated charging, contrary to ABE.

Lastly, destination drop might appear as an alternative to ABE that would require no support from the network. This alternative would consist in having the destination drop all packets that arrive too late, say after a transit deadline. However, it wastes network resources, since packets are dropped after being carried by the network, and the overall performance of such a scheme can become very poor [23].

3 The ABE Service

In this Section we define the ABE service, illustrate on an example its use by a multimedia application, and discuss how it could be introduced in the Internet. We discuss in detail the implication of the requirement *green does not hurt blue*. We identify and analyse the relationship with TCP-friendliness. This allows us to propose a set of router requirements, against which we propose to check implementations.

3.1 Definition of the ABE Service

ABE is an Internet service defined by the following set of characteristics.

1. ABE packets are marked either green or blue.
2. Green packets receive a low, bounded delay at every hop. Realistic values of the per-hop delay bound are discussed later in this section.
3. Applications are expected to control their rate in a TCP-friendly manner. An application is considered to send blue and green

packets which belong to the same flow, as opposed to two distinct flows.

4. *Green does not hurt blue:* If some source decides to mark some of its packets green rather than blue, then the quality of the service received by sources that mark all their packets blue remains the same or becomes better. This definition is deliberately non-specific. A formal one requires a longer argument which is done in Section 3.4.

5. All ABE packets belong to one single best effort class. If the total load is high, then every source may receive little throughput. However, entirely blue sources would experience more throughput than entirely green sources sharing the same network resources.

At very high bit rates, queueing delays are in general expected to be lower and high speed backbones probably will not need any delay differentiation. Hence, we currently expect ABE routers to be implemented at network peripherals, where bit rates are of the order of a few Mb/s (or even less for cellular radio systems). The value of the delay bound offered to the green service depends on how many hops are used by one flow. A multimedia flow probably uses a small number (2 to 6) of low speed hops. An interactive audio application has a delay budget of 100-150 msec, out of which 50 msec may be allocated to network delay. As a result, we expect the green per-hop delay bound to be set to a value in the range of 5 to 20 msec.

In today's Internet, it is considered as desirable to preserve packet ordering, though this is not always enforced. Similarly, an ABE node is expected to preserve packet order as much as possible; however, the delay preference given to green may result in a green packet overtaking a blue one.

In essence, ABE can be thought of as allowing an application to trade delay for throughput, by marking some packets green. In this context, characteristic 4, green does not hurt blue, is essential. In particular, it is desirable that an entirely blue flow receives at least as good average throughput as it would in a flat best-effort network i.e. if all packets were blue. If this is enforced, there is benefit to all: if some application decides to mark some packets green, then it must do so because it values the low delay more than a potential decrease in throughput; otherwise, it would mark the packets blue. In all cases, there

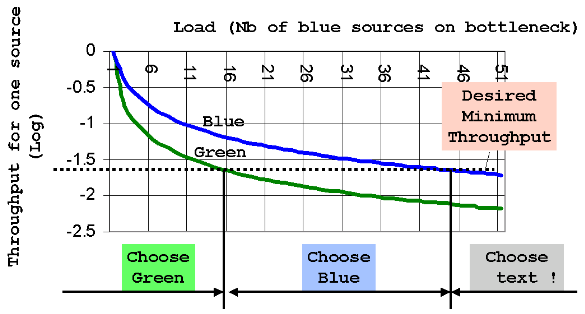

Figure 1: A possible strategy for a multimedia source using the ABE service.

is no penalty for other applications, which might choose to mark all their packets blue. Thus, it is not required to police the colour chosen by applications, provided they are TCP-friendly. The enforcement of friendliness [9, 8] is outside the scope of this paper. Thus ABE attempts to retain the simplicity of the original Internet single class, best effort service, while providing low delay to interactive applications. In particular, flat rate charging may be applied if an operator decides to do so.

3.2 An example

We discuss a very simple scenario, in order to illustrate how a source would use the ABE service.

Figure 1 shows a simple simulation where an interactive audio source competes with n background sources for one bottleneck. The source has the choice of marking packets blue or green. The other sources are all blue. Assume the source has a required minimum rate R_0 in order to function properly, for a given loss pattern in the network. The rate R_0 is shown by the horizontal dashed line. Also assume that the source is able to forward-correct packet losses, as long as the minimum rate is achieved (see [18] for such an application example; note that this would not be needed if ECN was used). The choice between green or blue is left to the audio application. It depends on its utility function $u(R, D)$, for a given throughput R and end-to-end network delay D. On this simplified example, we assume that the utility function for our source satisfies (1) $u(R, D) = 0$ for $R < R_0$ and (2) $u(R, D)$ is a decreasing function of D only for $R \geq R_0$. In other words, once a minimum rate R_0 is achieved which provides enough intelligibility, delay becomes the major impediment. For this source, the optimal strategy is to be green in the low load region, blue in the moderate load region, and to disconnect when the load is too high.

In the low load region, the source might have marked its packets blue, in which case it would have received more throughput, and, we assume, a larger delay. But because the source values low delay more than higher throughput (as long as $R \geq R_0$), it will not do so. This example illustrates that ABE opens up a new region of operation for the best-effort network: in low load scenarios, a source may decide to obtain less throughput at the benefit of low delay. In a flat best effort network, a network without the ABE service, there is no such option. Indeed, by refraining from sending at a higher rate, there is in general no impact on the queueing delay, because of external sources.

Note that this example is oversimplified. In general we expect more complex utility functions to be used. Note also that the detection of which region the source is currently operating in has to be made automatically by the source itself, using a colour adaptation algorithm. In companion work [21], we propose an ABE-aware audio application which combines colour adaptation, FEC control and TCP-friendly rate adaptation in order to maximise a utility function which depends both on delay and rate.

Unlike the multimedia source above, a source using TCP is probably more interested in its throughput and should thus mark all its packets blue. This follows immediately from the definition of the service. Intuitively, it is because ARQ protocols such as TCP are more sensitive to packet loss than to queueing delay, though queueing delay does have an impact.

An ABE aware source would probably use a colour mixing strategy, where they would send some green packets and some blue. This would for example be used by the colour adaptation algorithm for monitoring purposes. This is perfectly permissible and considered normal practice; in fact, apart from possibly policing TCP-friendliness, the network supporting ABE does not need to analyse individual flows. Source strategies would typically be performed at the application level as expected by Application Layer Framing (ALF) [21].

3.3 Inter-working and Migration: from Flat Best Effort to ABE

ABE could be used by an operator in two distinct ways; either as a separate service, or as a replacement to the flat (existing) best effort IP service. In this paper we focus on the latter. Indeed, as mentioned earlier, the initial thinking behind ABE was to provide support for interactive, adaptive Internet applications, while retaining the simplicity of the original Internet service model.

Replacing flat best effort by ABE requires a rule for assigning a colour to packets that do not have one (such packets come from a non-ABE source or network). By default, the rule is to assign a blue colour to packets. This is because of the characteristic that green does not hurt blue. Thus, ABE unaware sources receive the same service as they would if the network would be flat best effort. An operator might thus introduce ABE and let customers and other carriers gradually move to ABE, without any specific change to charging or control policies. To be more accurate, the requirement green does not hurt blue is intimately related with the issue of TCP-friendliness.

Conversely, consider an ABE aware source which uses a concatenation of networks, some ABE, some flat best effort. We have mentioned earlier that an ABE aware source probably has to implement a colour adaptation algorithm. Now, depending on traffic conditions, the ABE source might see small or large delays even for green traffic. This implies that the colour adaptation algorithm should not make any quantitative assumption about the value of end-to-end delay guaranteed for green traffic. Other than this, the interconnection of ABE and flat best effort networks poses no special requirement.

As mentioned earlier, ABE needs to be implemented only at network nodes where buffering delay may become large, larger than a threshold of the order of 5-20 msec. Thus, there is no need for ABE support in high speed backbones. ABE would typically be required at access nodes (modem or ADSL), at carrier interconnection points, and on cellular radio systems. An Internet draft [6] discusses how the ABE colour can be encoded in the IP header.

3.4 Green does not hurt Blue

In this section we define more accurately what it means that green does not hurt blue, as introduced in Section 3.1.

Intuitively, this requirement can be expressed by considering two scenarios; one where all sources are blue and the other where some sources decide, based on their own logic, to mark some packets green. The quality of service received by those packets which have remained blue in the second scenario should be as good as in the first one. This requires firstly that the delay is not any larger. Secondly, a packet which is not dropped (or marked with a congestion notification) in the first scenario would not be either in the second scenario. As a consequence, the throughput of an entirely blue flow would be at least as good in the second scenario (since we assume that flows are TCP friendly).

A problem with that simple, intuitive definition is that sources are assumed to be adaptive (TCP-friendly), thus in the second scenario, the packets sent by the sources will not be the same as in the first one. Furthermore, the behaviour of sources is dependent on their rate adaptation algorithm, which, while being TCP-friendly, may have a large number of different incarnations.

A first step in order to circumvent this difficulty is to introduce the following definition.

Definition 3.1 *(Local Transparency to Blue) Consider the scenario, flat best-effort, in which the node would forget the colour and thus treat all ABE packets as one single best effort class. An ABE node satisfies local transparency to blue if, for each packet that is blue in the original (ABE) scenario:*

1. *the delay is not larger in the real, ABE scenario than in the flat best-effort scenario*
2. *if the blue packet is not dropped (or marked with a congestion notification) in the flat best-effort scenario, then it is not either in the real, ABE scenario.*

It means that if some packets are marked green in a node it does not hurt blue packets, assuming one can ignore the effects due to rate adaptation in the sources. It is a necessary requirement to ensure green does not hurt blue. However, it may not be sufficient, since the rate adaptation algorithm at the source might produce a higher rate when the end-to-end delay is smaller.

Indeed, TCP-friendliness can be interpreted as follows. The source should produce a data rate not exceeding θ given by

$$\theta = \frac{s}{R\sqrt{\frac{2p}{3}} + 3t_1\sqrt{\frac{3p}{8}}p(1+32p^2)} \qquad (1)$$

where R is the round trip time, p the rate of loss events, t_1 the TCP retransmit time (roughly speaking, proportional to the round trip time), and s is the packet size [3].

Thus, it is quite possible that, by becoming green, a source would be allowed a higher data rate, due to the reduction in round trip time. Such a source would generate more packets than if it was blue, and there is the risk that, in some cases, it would hurt blue packets.

We should stress that the dependency of rate on round trip time in Equation (1) is not necessarily a desirable feature of a rate adaption algorithm. It should not be confused with the fact that a source using many hops should receive less throughput. This latter fact is desirable, but it is implemented by having a higher loss ratio. Further discussion on this is provided in [4]. Fixes have been suggested to rate adaptation algorithms, that would remove the dependency of rate on loss ratio [2]. If such fixes become widespread, then per-hop transparency to blue would indeed ensure that, from a throughput viewpoint, green does not hurt blue.

However, the current definition of TCP-friendliness does imply a dependency of rate on round trip time. In this context, it is necessary to compensate for the delay decrease obtained by green traffic. This leads us to the following requirement for an ABE node.

Definition 3.2 *(Throughput Transparency to Blue) Assume that sources employ a rate adaptation algorithm which conforms to a loss-throughput formula such as Equation (1). To provide blue with throughput transparency, the ABE node must ensure that an entirely green flow gets a lesser or equal throughput than if it were blue.*

In summary, we identified that supporting the green does not hurt blue requirement can be analysed as follows. A necessary condition is local transparency to blue (Definition 3.1). Given the bias against long RTTs in the TCP-friendly rate adaptation rules accepted today, ABE nodes have to additionally satisfy Definition 3.2, which, due to reasons we outlined, can only be achieved approximately. The solution to this issue is entwined with the very definition of TCP friendliness, and as such is beyond the scope of this paper. In Section 4, we propose an implementation that exactly satisfies Definition 3.1 and approximately satisfies Definition 3.2.

3.5 General router requirements

Following from the discussion in the previous section, a router implementing ABE must:

1. Provide low, bounded delay to green packets; the delay bound is fixed by network management, probably in the 5-20 msec range.

2. Provide local transparency to blue (Definition 3.1).

3. Provide throughput transparency to blue (Definition 3.2).

4. Minimise green packet dropping subject to the above requirements.

The first three requirements directly derive from the previous discussion. The last requirement is because an implementation should try to make the service as attractive as possible by keeping green packet loss low.

As mentioned earlier, we do not consider in this paper the task of enforcing TCP-friendliness. A per-flow implementation of ABE would be able to jointly support ABE and enforce TCP-friendliness. This is the object of ongoing work and is beyond the scope of this paper. Let us also recall here that, as mentioned in item 5 in Section 3.1, the relation between packet drop ratio and source rate should be enforced independently of packet colour.

4 A Router Implementation

In this Section we present a router implementation model to support the ABE service. This implementation assumes that the router has only output port queueing. It is based on a new scheduling concept, Duplicate Scheduling with Deadlines (DSD). We have also undertaken other implementations, based on different scheduling concepts, which include: a differential dropper based implementation in the ns2 simulator [23] and Linux kernel [5], and a dummy packet based implementation in the Dummynet emulator [5] .

We describe DSD, discuss its compliance to the ABE router requirements, as per Section 3.5, and show some experiment results from simulations.

Before delving into the details of the scheme's description, let us first explain its motivation.

One of the first schemes to implement ABE that comes to mind would probably be a FCFS scheduling discipline with a threshold drop policy to filter green packets. In that scheme, blue packets would be accepted until the buffer is full, whilst green packets will only be accepted if it can be served with no greater delay than some maximum d.

Whilst the scheme might operate "reasonably" well some of the time, most of the time there would be little or no incentive in being green. One wants to design a scheme that provides the best service possible to green while still ensuring throughput transparency, i.e blue flows receive at least as much throughput as they would receive given a flat best-effort service. Any *significant* extra gain by blue packets is at the expense of green ones.

The gain blue packets would enjoy under ABE should be kept to a minimum such that there is still an incentive to use green packets whenever appropriate.

We formalise this by the following optimisation problem. We wish to minimise the number of green losses subject to the following constraints:

- Green packets receive a no larger queueing delay than d.

- Local transparency to blue (Definition 3.1) holds.

- The scheduling is work conserving.
- No reordering: Blue (respectively green) packets are served in the order of arrival.

The provision of throughput transparency is discussed in Section 4.2. An algorithm which is a solution to this problem is described in the next section. It is based on a new scheduling concept called *duplicates*, which solves this problem by approximating the operation (behaviour) of a flat best-effort service.

4.1 Duplicate Scheduling with Deadlines (DSD)

An example of how DSD works is given in Figure 2. Central to this scheme is the concept of a *virtual queue*, which is fed with *duplicates* of all incoming packets which are served by a FCFS discipline with rate c, as they would be in a flat best-effort. The times at which the duplicates are served is used to assign blue packets *deadlines* at which they would have (approximately) been served in a flat best-effort service. Actual blue and green packets are fed into two separate queues. Blue packets are always served at the latest their deadline permits subject to work conservation. Green packets are served in the meantime if they have been in the queue for less than d seconds, and are dropped otherwise.

All packets are duplicated and fed in a virtual queue, served at rate c, and with buffer size *Buff*. If the virtual queue length has reached *Buff*, the duplicate and the original blue packet are dropped. Otherwise, the duplicate is enqueued in the virtual queue, and the original packet is tagged with a deadline, which is the time at which the duplicate will be served in the virtual queue. The original packets are enqueued in two separate queues, one for greens and one for blues. Blue packets are always served no later than their deadlines. If at a given time, the blue packet at the head of the queue does not exceed its deadline by letting the first green packet go first, the latter packet is served – provided it waited for less than d seconds. Green packets that have to wait for more than d seconds are dropped from the green queue.

Upon the arrival of a packet at the output port, the algorithm is summarised below:

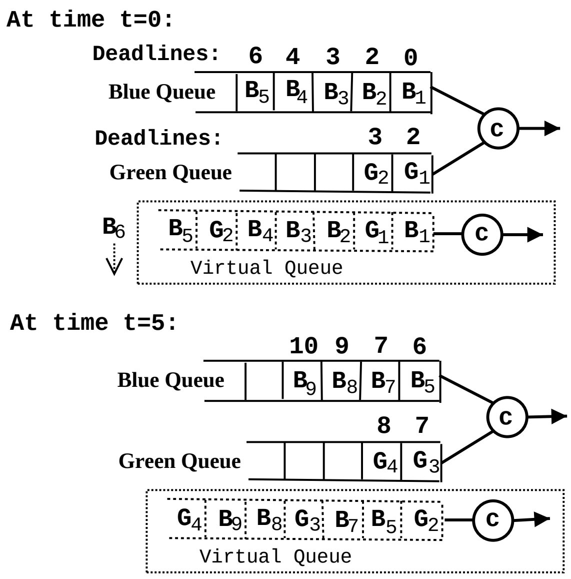

Figure 2: Two snapshots as an example of DSD, at time $t = 0$ (top) and $t = 5$ (bottom). For this example, all packets are the same size and "packet" time is used. The maximal buffer size is $Buff = 7$ packets. The maximum green queue wait is $d = 3$ packets. B and G denote blue and green packets respectively. In the first snapshot, B_1 is served at time $t = 0$ in order to meet its deadline, then G_1, B_2, B_3, B_4. G_2 has to be dropped from the green queue because it has to wait for more than $d = 3$, whereas B_6 had to be dropped because the virtual queue length was $Buff$ when it arrived. At time $t = 5$, we reach the situation of the second snapshot. As no blue packet has reached its deadline yet, G_3 can be served, followed by B_5, B_7, G_4, B_8, and B_9.

```
Packet enqueueing Algorithm:
  add to virtual flat best-effort queue

  if blue:
    if dropped in virtual flat queue
      drop
    else
      vd = queueing delay received in virtual queue
      maxServiceTime = now + vd
      tag maxServiceTime to packet
      add to blue queue
    else if green:
      maxServiceTime = now + d
      tag maxServiceTime to packet
      add to green queue
```

```
Packet Serving Algorithm (without control loop):
  remove all green packets whose maxServiceTime
  cannot be met

  if no green packet to serve:
    serve blue packet if any
  else if no blue packet to serve:
    serve green packet
  else:
    pg = transmission delay for
     head of green queue
    mstb = maxServiceTime of head of blue queue
    if now <= mstb - pg
      serve green packet
    else
      serve blue packet
```

We have favoured clarity in description over efficiency. Also, we have not mandated that the virtual queue employ drop-tail queueing although in the results we show here it is. An Active Queue Management scheme such as RED [10] can be supported for blue traffic by applying it to the virtual queue, and using those results in assigning losses and deadlines.

Compliance Let us now look at its compliance with the ABE router requirements:

1. Low bounded (per hop) delay for the green packets is enforced by dropping a green packet that cannot be served in the deadline d.

2. Local transparency to blues (Definition 3.1) is ensured through dropping blues only when they would be in the flat best-effort, and in not letting accepted blues wait longer in the queue than they would in flat best-effort.

3. The DSD algorithm described so far does not ensure throughput transparency to blue (Definition 3.2), which depends on the TCP feedback. This is done by the control loop described in the next section.

4. Minimising green packet dropping is achieved optimally by the packet enqueueing and serving algorithms.

4.2 Control loop for DSD

The duplicate scheme, as it stands, provides the optimal performance to green while constrained to support *local transparency to blue.* In this section, it is enhanced with a control mechanism in order to provide *throughput transparency to blue.*

When TCP feedback comes into play, the throughput of blue and green can be affected by, either violating throughput transparency and not providing blue with enough throughput, or providing green flows with not sufficient throughput for ABE to be attractive.

To cater for these two scenarios, we add a control loop which adjusts to the rate-adaptive nature of TCP. Since TCP-friendly sources are sensitive to delay, we correct for any advantage green packets might receive by increasing their delay. DSD, as described above, in addition to providing minimum green losses for local transparency, also minimises the delay green packets receive. This is because in the event of either a green or a blue being able to wait and still meet their deadlines, i.e. if both packets can wait, we always serve the green packet first. This is not a requirement of ABE, and we can still maintain local transparency, and not systematically favour the green in this scenario. Note that by increasing the delay for green, we are reducing their throughput, thus restoring throughput transparency.

We generalise the packet serving algorithm by introducing a *green bias* g in the range $[0, 1]$, which determines the extent to which we favour green over blue when both the first blue and green packets have not yet reached their deadlines. More precisely, when both the blue and green packets can wait, g is the probability that we serve the green packet first. The value $g = 1$ corresponds to the algorithm of the previous subsection, where the green packet is always served if both the blue and green packets can wait. Conversely the value $g = 0$ corresponds to the systematic serving of the blue packet first. In the example in Figure 2, the packets served would have thus been, successively, B_1, B_2, G_1, B_3, B_4, B_5, B_7, G_3, G_4, B_8 and B_9.

We use g as a control parameter to balance the throughputs of green and blue. These are estimated from Equation (1). Let $\theta_b(t)$ and $\theta_g(t)$ be these estimates for the blue and the green respectively at time t. The target of the control is to maintain their ratio to a desired value γ, which is slightly larger than one, to provide blues with a small advantage in throughput, and offer a safety margin to protect from errors in the estimation of the throughputs. We derived a control loop equation for the green bias

$$g(t+1) = (1-\alpha)g(t) + \frac{\alpha}{1 + (\gamma\theta_g(t)/\theta_b(t))^K},$$

where α is the adaptation gain ($0 < \alpha < 1$), and $K > 0$ is a parameter that shapes the nonlinear function of θ_g/θ_b. When θ_b is close to $\gamma\theta_g$, the control law drives $g(t)$ towards $1/2$, which amounts to serving both queues without bias. It pushes $g(t)$ towards 1 if $\theta_b > \gamma\theta_g$ and towards 0 otherwise.

This algorithm is only one possible scheme, and its performance can be improved, for example, by taking into account the deadlines of all the packets in the queues, and not just those at the head. Such extensions are the subject of further investigation.

Let us just mention here the extension used to cope with a very lightly loaded network, in which case the green may suffer a few losses because of the bounded delay, while the blue have practically no losses at all. As a result of the TCP feedback mechanism, the green sources decrease their throughput much more often than the blue, at such a pace that the throughput of the blue can exceed significantly the throughput of the green, even when $g = 1$. In this case, we can only increase green throughput by relaxing partly the local transparency requirements. We preserve the delay transparency, but instead of always serving a blue packet when its deadline would otherwise not be met, we will drop it with probability ϵ, where $\epsilon = 0$ for $g < 1$. We use ϵ as a second control parameter, derived in a similar fashion to $g(t)$ above.

The serving algorithm is thus modified as follows:

```
Packet Serving Algorithm (with control loop):
  remove all green packets whose
  maxServiceTime cannot be met

  if no green packet to serve:
    serve blue packet if any
  else if no blue packet to serve:
    serve green packet
  else:
    pg = transm. delay for head of green queue
    mstb = maxServiceTime of head of blue queue
    pb = transm. delay for head of blue queue
    mstg = maxServiceTime of head of blue queue

    if now > mstb - pg   /* blue cannot wait */
  drop blue with probability $\epsilon$
    if blue not dropped:
      if now > mstg - pb /* green cannot wait */
        serve green
      else with probablity g(t)
        serve green
      else
    serve blue
```

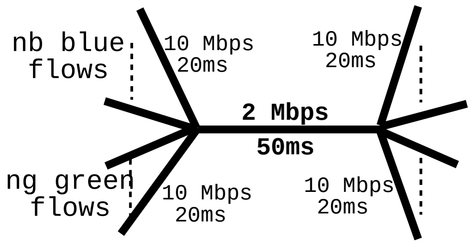

Figure 3: Simulation topology.

4.3 Simulation Results

In this section, we provide illustrative simulations of the implementation using the simple topology, shown in Figure 3. For brevity, we omit extensive simulations, which showed that under many varied environments, the service requirements are met.

There are n_b blue and n_g green sources which are TCP Reno with always a packet to send (greedy). We expect green sources to use a rate-adaptive application which is TCP-friendly, but for direct comparison we use TCP in this case.

The router buffer size was 45 packets (i.e. $Buff = 45$) and the maximum delay to green d is $0.04s$. All packets are fixed with size 1000 bytes.

We first look at the case where there are $n_b = 5$ blue and $n_g = 5$ green flows. Figure 4 shows the average number of packets received by each blue and green connection at each time t. Figure 5 shows the end-to-end delay distributions received for green packets under ABE and flat best-effort.

It can be seen from the results that the ABE router requirements were satisfied. Green queueing delay is small and bounded by $d = 0.04s$, satisfying requirement 1. With this traffic load it was not required to be weaken transparency to help green ($\epsilon = 0$ was used), and thus requirement 2 of local transparency to blue was satisfied. The blue flows receive at least as much as they would as with flat best-effort, satisfying requirement 3. They actually receive more, thus receiving benefit

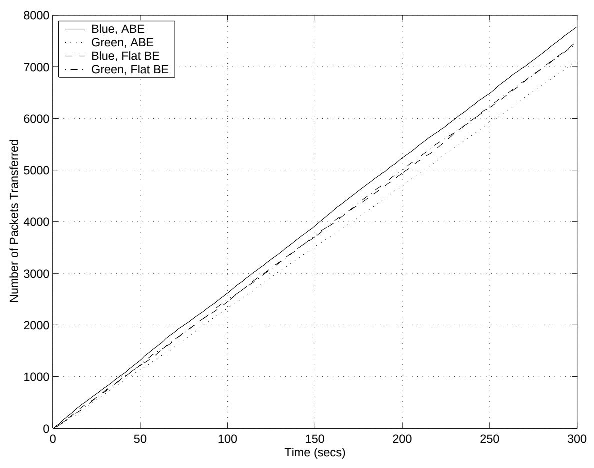

Figure 4: Average number of packets transferred per green and blue connection, as a function of time t, when the router implemented ABE/DSD and when it implemented flat best-effort. The results are obtained by simulating the network described on Figure 3, with 5 blue and 5 green flows.

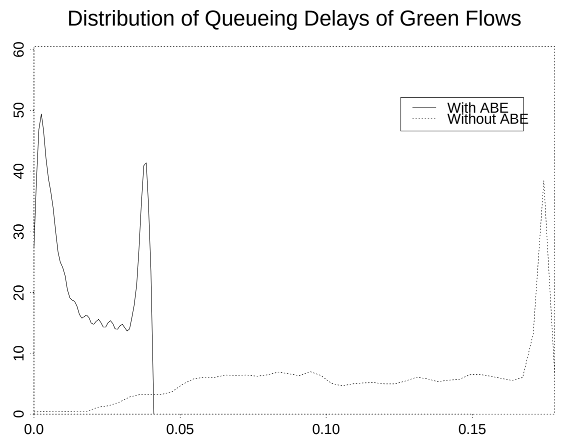

Figure 5: Queueing Delay distributions received for green packets under ABE/DSD and flat best-effort. 5 blue and 5 green flows.

from the use of ABE. The green sources receive less throughput but a reasonable amount which satisfies requirement 4.

Of course, the same number of blue and green sources does not occur in a general, and ABE is designed to work independently of the asymmetry in green and blue traffic. We now look at such cases. Case (i) is where blue traffic dominates $n_b = 5$, $n_g = 2$, and case (ii) is where green traffic dominates $n_b = 2$, $n_g = 5$. Throughput for blue and green is shown in Figure 6 and the delay distribution for green is shown in Figure 7. In all cases, it can be seen that all router requirements are satisfied.

5 Conclusions

We have described ABE, a new service which enables best-effort traffic to experience a low delay, at the expense of possibly more throughput. ABE is targeted at supporting rate-adaptive multimedia applications, with no concept of reservation or signalling and while retaining the spirit of a flat rate network. 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. ABE allows a collection of rate-adaptive multimedia applications to drive the network into a region of moderately high load and low delay. It also allows such an application to trade reduced throughput for low delay, thus in some cases increasing its utility. The design of a multimedia adaptive application that would exploit the new degree of freedom offered by ABE can be found in [21].

It should be stressed that ABE is a new service in its own right and not a substitute for reservation or priority services. In contrast, with ABE, both delay sensitive (green) and throughput sensitive (blue) traffic share the same resources, and high load in any of the two pools affects the other. We proposed to introduce ABE as a replacement for the existing best effort Internet service.

We have defined the ABE service, its requirements and properties. We also addressed deployment issues. In addition, we have presented a router implementations and discussed its compliance. Our simulation results show the benefits of the new degree of freedom offered by ABE in the best-effort services. We have found that under ABE, blue packets received more throughput than under a flat best-effort network while giving a low bounded delay to green packets.

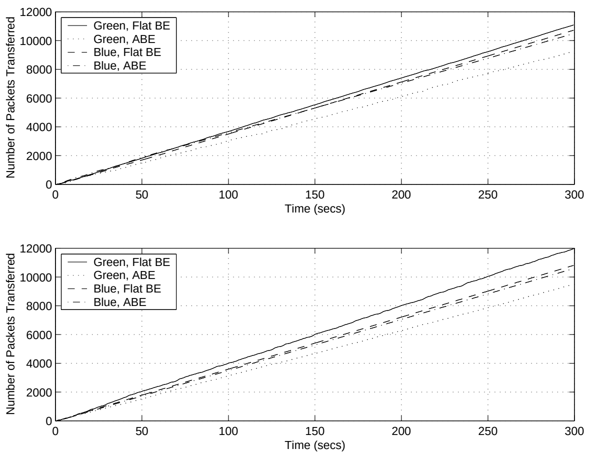

Figure 6: Average number of packets transferred per green and blue connection, as a function of time t, when the router implemented ABE/DSD and when it implemented flat best-effort. The results are obtained by simulating the network described on Figure 3, with 5 blue and 2 green in the first plot and 2 blue and 5 green in the second one.

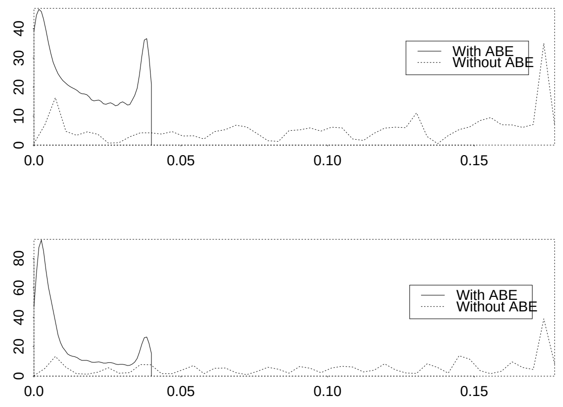

Figure 7: Queueing Delay distributions received for green packets under ABE/DSD and flat best-effort. 5 blue and 2 green in the first plot, and 2 blue and 5 green in the second.

References

[1] J. Crowcroft. All you need is just 1 bit. Keynote Presentation. *IFIP Conf. on Protocols for High Speed Networks*, Oct. 1996.

[2] Henderson, T.R., E. Sahouria, S. McCanne, and R.H. Katz. Improving Fairness of TCP Congestion Avoidance. *Proceedings of IEEE Globecom '98, Sydney, Australia, Nov. 1998*

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

[4] M. Vojnovic, J. Y. Le Boudec, C. Boutremans. Global Fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times. Proceedings of IEEE INFOCOMM'2000, Tel Aviv, Israel, March 2000.

[5] ABE Project Web Page. http://ica1www.epfl.ch/PS_files/abe.htm

[6] The Alternative Best-Effort Service. Internet Draft, Work in Progress. draft-hurley-alternative-best-effort-00.txt

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

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

[9] B. Suter, T.V. Lakshman, D. Stiliadis, A. Choudhury. Design Considerations for Supporting TCP with Per-flow Queueing. *Proceedings of IEEE INFOCOM 98.*

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

[12] M. May, J. Bolot, C. Diot, A. Jean-Marie. 1-bit Schemes for Service Discrimination in the Internet: Analysis and Evaluation. *Technical Report no 3238*, Inria.

[13] Y. Moret, S. Fdida. A Proportional Queue Control Mechanism to Provide Differentiated Services. *International Symposium on Computer Systems*, Belek, Turkey, October 1998.

[14] V. Jacobson, K. Nichols and K. Poduri. An Expedited Forwarding PHB. RFC2598.

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

[16] C. Dovrolis, D. Stiliadia, P. Ramanathan Proportional Differentiated Services: Delay Differentiation and Packet Scheduling Proceedings of ACM SIGCOMM'99.

[17] Floyd, S. TCP and Explicit Congestion Notification. *ACM Computer Communication Review.* V. 24 N. 5, October 1994, p. 10-23.

[18] J. Bolot, S. Fosse-Parisis, D. Towsley. Adaptive FEC-Based Error Control for Interactive Audio in the Internet. *Proceedings of IEEE Infocom 99.*

[19] C. Diot, C. Huitema, T. Turletti. Multimedia Application should be Adaptive. *HPCS*, Aug. 1995.

[20] K. Kilkki, Jussi Ruutu. Simple Integrated Media Access - an Internet Service Based on Priorities 6th International Conference on Telecommunication Systems, 1998.

[21] C. Boutremans and J.Y. Le Boudec. Adaptive delay aware error control for internet telephony. Technical Report Research Report DSC 200031, EPFL-DSC, http://dscwww.epfl.ch, 2000.

[22] C. Huitema. Quality today in the internet. ftp.telecordia.com/pub/huitema/stats/quality.today.html.

[23] Paul Hurley, J.Y. Le Boudec, and P.Thiran. The alternative best-effort service. Technical Report Research Report DSC1999/036, EPFL-DSC, http://dscwww.epfl.ch, 1999.