ABE: Providing a Low-Delay Service within Best Effort

Paul Hurley and Jean-Yves Le Boudec, EPFL
Patrick Thiran, Sprint ATL/EPFL
Mourad Kara, University of Leeds

Abstract

We propose alternative best effort (ABE), a novel service for IP networks, which relies on the idea of providing low delay at the expense of maybe less throughput. The objective is to retain the simplicity of the original Internet single-class best-effort service while providing low delay to interactive adaptive applications. With ABE, 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 color is made 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 large enough throughput. In contrast, an application that transfers binary data such as bulk data transfer will seek to minimize overall transfer time and send blue traffic. We propose router requirements that aim at enforcing benefits for all types of traffic, namely that green traffic achieves low delay and blue traffic receives at least as much throughput as it would in a flat (legacy) best effort network. ABE is different from differentiated or integrated services in that neither packet color can be said to receive better treatment; thus, flat rate pricing may be maintained, and there is no need for reservations or profiles. In this article 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. We propose and analyze an implementation based on a new scheduling method called duplicate scheduling with deadlines. It supports any mixture of TCP, TCP-friendly, and non-TCP-friendly traffic.

We present an enhancement to the IP best effort service, alternative best effort (ABE), which relies on the idea of providing low delay at the expense of maybe less throughput. The motivation for such a service is twofold. First, there now exist interactive multimedia applications that perform well across a wide range of loss and throughput conditions [1, 2], but for which delay often remains the major impediment [3]. Second, unlike differentiated services, we would like to design a service where it is not required to police how much traffic uses the low delay capability, in order to retain the operational simplicity of a single-class network.

The article is organized as follows. We define the ABE service and analyze its implications. In particular, we identify and discuss the central issue, called *green does not hurt blue*. We also discuss migration issues from the traditional IP service (flat best effort) to ABE. As a proof of concept, a router implementation is described; it is based on the combination of a scheduler called *duplicate scheduling with deadlines* (DSD) and a traditional control loop. Implementations were done in the Linux kernel, in the Dummynet network emulator, and in the ns-2 simulator [4]. Simulation results of the ns-2 implementation are shown. We review related work and position ABE with respect to other proposals in differentiated services.

The ABE Service

Definition of the Service

ABE is defined as follows:

- ABE packets are marked either green or blue.[1]
- Green packets receive a low bounded delay at every hop. Realistic values of the per-hop delay bound are discussed later in this section.
- *Green does not hurt blue*: If some source decides to mark some of its packets green rather than blue, the quality of the service(delay and throughput) received by sources that mark all their packets blue remains the same or becomes better. This definition is made more specific later.
- All ABE packets belong to one single best effort class. If the total load is high, every source may receive little throughput. However, entirely blue sources experience more throughput than entirely green sources sharing the same network resources.

[1] *The choice of the terms blue and green, two neutral colors, is to indicate that neither of the two has priority over the other, while green, the color of the traffic light signal for go, indicates low queuing delay.*

0890-8044/01/$10.00 © 2001 IEEE

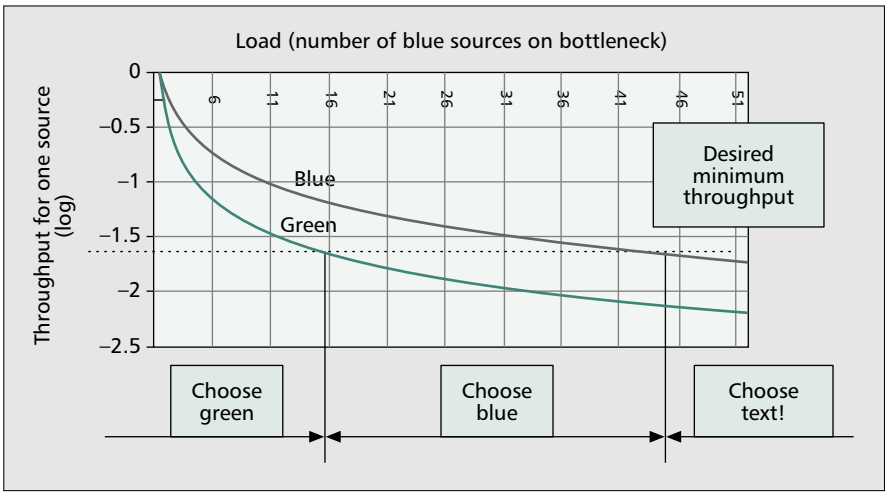


Figure 1. *A possible strategy for a multimedia source using the ABE service.*

A consequence of these requirements is that green packets are more likely to be dropped during bouts of congestion than blue packets, or, if explicit congestion notification (ECN) [5] is used, to be marked with the congestion bit. ECN provides congestion feedback to the source by marking a bit in the packet header, enabling it to adjust to feedback without necessarily dropping its packets. For simplicity, in the rest of the article we consider only non-ECN-capable systems.

In essence, ABE can be thought of as allowing an application to trade delay for loss or less throughput by marking some packets green. The third requirement, green does not hurt blue, derives from the objective that the color chosen by an application need not be policed. Indeed, if the third requirement is enforced, an application that decides to mark some packets green must do so because it values the low delay more than a potential increase in loss (or decrease in throughput); otherwise, it would mark its packets blue. In all cases, there is no penalty for other applications, which might choose to mark all their packets blue. This requirement also plays a role in interworking and migration (discussed later). Note that ABE supports traffic that may be only TCP-friendly or non-TCP-friendly, or a mixture of the two.

In Fig. 1 we illustrate how a TCP-friendly source would use the ABE service, by showing a simple simulation where an interactive adaptive audio source competes with n background sources for one bottleneck. The source has the choice of marking packets blue or green. 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 [1] 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 $R > R_0$. 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. Note that this example is oversimplified. In general we expect more complex utility functions to be used; see [6] for an actual audio source using such a utility function.

An ABE-aware source would probably use a color mixing strategy, where they would send some green packets and some blue. This would be used, for example, by a color adaptation algorithm that sends probe packets of either color in order to determine which region the source is currently operating in. This is perfectly permissible and considered normal practice. In fact, apart from possibly policing TCP friendliness if so required [7, 8], a network supporting ABE does not need to analyze individual flows. Unlike the multimedia source above, a source using TCP is probably more interested in its throughput and should thus mark all its packets blue. Intuitively, it is because automatic repeat request (ARQ) protocols such as TCP are more sensitive to packet loss than queuing delay, although queuing delay does have an impact (see a later section for an illustration).

The Internet Engineering Task Force (IETF) mandates that non-TCP sources be TCP-friendly [10]; namely, the source should not receive more throughput than a TCP flow would. This is for reasons of fairness and to avoid congestion collapses. However, it is still the case that many multimedia flows are *not* TCP-friendly. Thus, the requirement that green does not hurt blue applies even if green traffic originates from non-TCP-friendly sources. Note that non-TCP-friendly sources may, in some cases, severely hurt other TCP-friendly sources, and this is true with or without ABE. The requirement simply means that giving low delay to such sources does not make things worse. A later section illustrates by simulation how an implementation based on DSD supports non-TCP-friendly sources.

At very high bit rates, queuing delays are in general expected to be low, 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 on the order of a few megabits per second (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 (2–6) of low-speed hops. An interactive audio application has a delay budget of 100–150 ms, out of which 50 ms 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 5–20 ms.

Green Does Not Hurt Blue

In this section we define more accurately the service requirement that green does not hurt blue, introduced earlier. We subdivide the requirement into two parts. The first part addresses the case of non-TCP-friendly sources:

Definition 1 — Local transparency to blue — Consider the scenario of flat best effort, in which a node would forget the color and thus treat all ABE packets as one single best effort class. The node satisfies local transparency to blue if, for each packet that is blue in the original (ABE) scenario:

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

Later we describe DSD, which provides local transparency to blue.

Now if some sources sending green traffic are TCP-friendly and greedy, local transparency to blue may not be sufficient. Indeed, a common interpretation of TCP friendliness is that the source data rate should not exceed θ given by

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

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 [10]. Thus, it is quite possible that, by becoming green, a TCP-friendly 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. This leads to the second part of the requirement.

Definition 2 — Throughput transparency to blue — Assume that sources employ a rate adaptation algorithm which conforms to a loss-throughput formula such as Eq. 1. To provide blue with throughput transparency, the ABE node should ensure that an entirely green flow gets a lesser or equal throughput than if it were blue.

Unlike local transparency, throughput transparency seems impossible to implement exactly: On the one hand, it requires knowing the round trip time for every flow, which is not feasible in practice. On the other hand, the rate adaptation algorithm implemented by a source may significantly deviate from a straight application of Eq. 1. Indeed, the dependence of rate on round-trip time in Eq. 1 is not necessarily a desirable feature of a rate adaptation 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 is implemented by having a higher loss ratio. Further discussion on this is provided in [11]. Fixes have been suggested to rate adaptation algorithms that would remove the dependence of rate on round-trip time [12]. If such fixes were to become widespread, throughput transparency to blue would be an automatic consequence of local transparency to blue (which can be exactly implemented, e.g., with DSD). Thus, we consider the requirement for throughput transparency to be loose.

Our approach to providing throughput transparency to blue uses a controller that acts on a parameter g of the DSD scheduler, which controls the service received by green packets. g is a factor that determines, in case of a tie between green and blue, the probability of forwarding a green packet. The delay and loss ratio are monitored, and using Eq. 1, the controller adjusts g to make sure throughput transparency is maintained. The controller solves the issue of evaluating the round-trip time by observing that an underevaluation of green round-trip times is conservative for blues. Thus, the controller assumes that all flows are greedy and have a total round-trip time equal to the queuing time at this node plus a fixed virtual base value (20 ms in the implementation). This value is kept small to make it unlikely that the real value could be below it. A potential problem could be that the drop probability becomes higher than necessary for either green flows with higher round-trip times, nongreedy green flows, or nonadaptive green flows. However, our simulations indicate that the combination of the DSD scheduler and controller does not seem to produce that problem.

An alternative coarse implementation may consist of avoiding any single green flow from getting too large a fraction of throughput by examining the drop history for greens, as proposed in [8]. Researching such an alternative is a subject of future work.

Router Requirements

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

- Provide low bounded delay to green packets; the delay bound is fixed by network management, probably in the 5–20 ms range.
- Provide local transparency to blue (definition 1).
- Provide throughput transparency to blue (definition 2).
- Preserve packet sequence within blue and within green.
- Keep green packet loss as low as possible while adhering to the above requirements.

The first three requirements directly derive from the previous discussion. The fourth requirement is because an implementation should try to make the use of green in the service as attractive as possible.

In today's Internet, it is considered desirable to preserve packet ordering, although 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. It is desirable that any ABE implementation is also work-conserving, although this is not a strict requirement.

Interworking and Migration

ABE may be used by an operator in two distinct ways: either as a separate service, or as a replacement of the flat (existing) best effort IP service. In this article we focus on the latter. Replacing flat best effort with ABE requires a rule for assigning a color to packets that do not have one (such packets come from a non-ABE source or network). The default is to assign blue to packets. Indeed, because of the characteristic that green does not hurt blue, ABE-unaware sources receive the same service as they would if the network were 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.

Conversely, consider an ABE-aware source that 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 color adaptation algorithm. Now, depending on traffic conditions, the ABE source might see small or large delays, even for green traffic. This implies that the color adaptation algorithm should not make any quantitative assumption about the value of end-to-end delay guaranteed for green traffic. Reference [13] discusses how the ABE color can be encoded in the IP packet header.

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 queuing. It is based on a new scheduling concept, DSD. We have also undertaken other implementations based on different scheduling concepts, which include a differential-dropper-based implementation in the ns2 simulator [14] (first outlined in [15]) and in the Linux kernel [16], and a dummy-packet-based implementation in the Dummynet emulator [16]. We describe DSD, discuss its compliance with the ABE router requirements as seen earlier, and show some experiment results from simulations.

Before delving into the details of the scheme's description, its motivation is first explained. One of the first schemes to implement ABE that might spring to mind is a first come first served (FCFS) scheduling discipline with a threshold drop policy to filter green packets. In such a scheme, blue packets would be accepted when the buffer is not full, while green

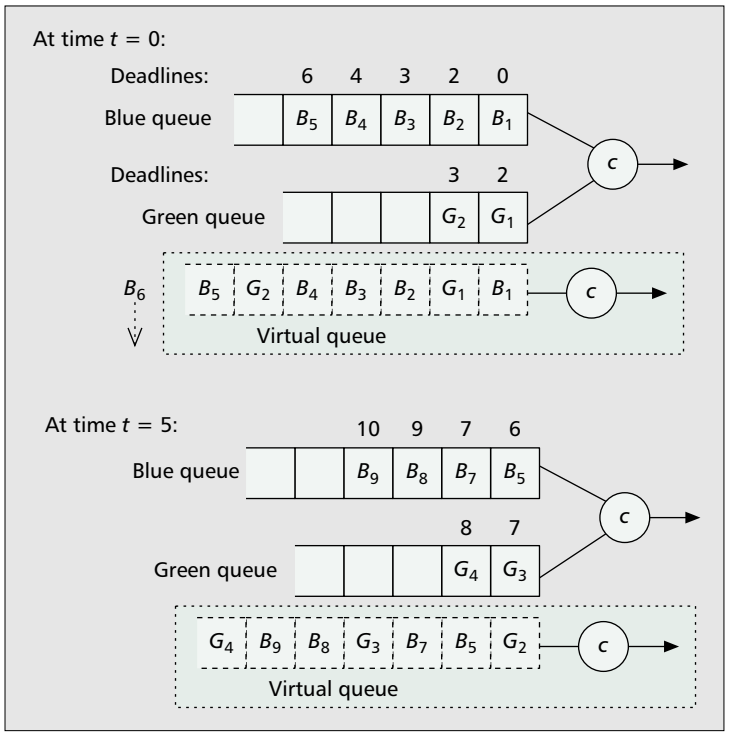
packets would only be accepted if they can be served with a delay no greater than some maximum d.

Most of the time, though, there would be little or no incentive to be green. What is desired is to provide green with the best service possible while still ensuring that green does not hurt blue. 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. This can be formalized by the following optimization problem: minimize the number of green losses subject to the following constraints:

- Green packets receive a queuing delay no larger than d.
- Local transparency to blue (definition 1) holds.
- The scheduling is work-conserving.
- No reordering: blue (respectively green) packets are served in the order of arrival.

A solution to this problem is the DSD, a new scheduling algorithm based on the concept of *duplicates*. Deadlines are assigned to packets as they arrive, green and blue packets are queued separately, and the deadlines of the packets at the head of blue and green queues are used to determine which is to be served next.

As previously discussed, throughput transparency as well as local transparency is required for ABE to ensure rate-adaptive green flows do not hurt blue. This is facilitated by the use of a parameter g, which is used in deciding which queue should be served in the event that the deadlines of the packets at the head of each queue can both be met if the other queue was served beforehand. The value of g used at any given time is determined by a control loop as described later. We can now describe DSD in detail.



■ Figure 2. *Two snapshots as an example of DSD, at time* t = *0 (top) and* t = *5 (bottom).*

Duplicate Scheduling with Deadlines (DSD)

Duplicates of all incoming packets are sent to a virtual queue with buffer size *Buff*. A duplicate is admitted if the virtual buffer is not full. Packets in the virtual queue are served according to FCFS at rate c, as they would be in flat best effort. The times at which duplicates will be served are used to assign blue packets deadlines at which they would have (approximately) been served in flat best effort. The original arriving packets are fed according to their color into a green and a blue queue. 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 s, and dropped otherwise.

A blue packet is dropped if its duplicate was not accepted in the virtual queue. Otherwise, it is tagged with a deadline, given by the time at which its duplicate will be served in the virtual queue, and placed at the back of the blue queue.

A green packet is accepted if it passes what is called the *green acceptance test* and dropped otherwise. A green packet arriving at time t fails the test if the sum of the length of the green queue at time t (including this packet), and of the length of the first part of the blue queue that contains packets tagged with a deadline less than or equal to $t + d + pg_{\text{new}}$, where pg_{new} is the transmission delay for the incoming green packet, is more than $c\ (d + pg_{\text{new}})$, and passes otherwise. The use of the test ensures the total buffer occupancy, namely the sum of the green and blue queue lengths, does not exceed *Buff*, which is discussed later.

An example of how DSD works is given in Fig. 2. For this example, all packets are the same size, and "packet" time is used. To facilitate understanding, we consider first the case where green packets do not undergo the green acceptance test and where $g = 1$. 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, and 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. Since 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.

Consider again the example in Fig. 2, except green packets are now enqueued only if they pass the green acceptance test. This amounts here to accepting a green packet at time t if the number of green packets in the queue at time t, augmented by the number of blue packets in the queue with a deadline between $[t, t + 4]$, is no more than 4. The only difference from Fig. 2 is that G_2 is no longer enqueued. Indeed, when it arrived, the green queue already contained packet G_1, and the blue queue contained packets B_1, B_2, and B_3. The total queue length at time t was 5 packets (including G_2), so G_2 fails the test.

An accepted green packet is then assigned a deadline which is the sum of its arrival time plus its maximum waiting time d, and placed at the back of the green queue. At each service time, a decision is made as to which queue to serve. The serving mechanism's primary function is to ensure that blue packets are always served no later than their deadlines. The best performance green could receive would be to then serve the green queue as much as possible, subject to this restriction. However, as previously discussed, in addition to local transparency, throughput transparency is needed to ensure that green adaptive applications do not benefit too much from lower delay.

It can happen at service time that both blue and green packets at the head of their respective queues are able to wait, since letting the other packet go first would still allow it to be served within its deadline. For the purpose of supporting throughput transparency, when this situation arises the packet serving algorithm uses the current value of the *green bias* g, a value in the range [0,1], to determine the extent to

which green is favored over blue. More precisely, when both blue and green packets can wait, g is the probability that the green packet is served first. The value $g = 1$ corresponds to the case where green is always favored. Conversely, the value $g = 0$ corresponds to the systematic favoring of blue packets. In Fig. 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.

A value of g less than 1 causes the delay for green traffic to be increased. This increase in delay for green TCP-friendly traffic reduces their throughput, thereby enabling blue traffic to increase its throughput. Increasing the delay of non-TCP-friendly traffic may not reduce their throughput, but blue flows are, in the worst case, equally as protected from this type of traffic as in a flat best effort service. The value of g choice is made according to a control loop, described later.

All green packets who miss their deadline by waiting for more than d seconds (these packets are said to have become *stale*) are removed from the green queue. At service time, the possible events that arise and packets served by DSD are shown in Table 1.

Pseudocode for DSD is given below. Let *now* be the current time, p.deadline denote the latest time a packet p can remain in the queue (whose value is tagged onto packet p), and p.transmissionDelay denote its transmission delay.

Packet Enqueuing Algorithm

```
packet p arrives at the output port
dup = p
Add dup to the virtual queue

if p is blue
   if dup was dropped from virtual queue
     drop p
else
   vd = queuing delay received by dup
        in virtual queue
   p.deadline = now + vd
   add p to blue queue
else // p is green
   if p fails "green acceptance test"
     drop p
   else
     p.deadline now + d
     add p to green queue
```

Packet Serving Algorithm

```
drop stale green packets, those packets from
green queue who cannot be served within their
deadline

headGreen = packet at head of green queue
headBlue = packet at head of blue queue

if headGreen 0 // no green to serve
   if headBlue !0
     serve headBlue
else if headBlue 0 // no blue to serve
   serve headGreen
else // both queues contain packets
   p_g headGreen.transmissionDelay
   dead_g headGreen.deadline
   p_b headBlue.transmissionDelay
   dead_b headBlue.deadline

   if now > dead_b – p_g
     serve headBlue // because it cannot wait
   else if now > deadg – p_b
     serve headGreen // because it cannot wait
   else with probability g
     serve headGreen
   else
     serve headBlue

"green acceptance test"
  pg_new transmission delay for p
  l_g length of green queue
  l_b length of packets in blue queue with
      deadlines < now + d + pg_new

  if l_g + l_b > c * d + pg_new)
    return "p fails test"
  else
    return "p passes test"
```

It is not mandated that the virtual queue employ drop-tail queuing, although in the simulation results shown it is. An active queue management scheme such as random early detection (RED) [17] can be supported for blue traffic by applying it to the virtual queue, and using those results in assigning losses and deadlines.

Removing stale green packets, those packets from the green queue whose deadline cannot be met, involves a search of this queue up to the first alive green packet. In practice, these stale green packets can be cleaned up between service times, as was done in our dummynet implementation, and it has proven sufficiently fast. However, for really high-speed networks this search may prove expensive. As such, further optimizations of this algorithm may be needed, and are the subject of ongoing work.

Some of the building blocks in DSD are similar to those in other scheduling techniques. The calculation and tagging of deadlines to each arriving packet is also performed by earliest deadline first (EDF)[18] schedulers and variants. However, EDF sorts packets according to deadlines, whereas DSD remains first in first out (FIFO) within each of its two queues, and the deadlines are used at service to determine whether the head of the green or blue queue should be served. The use of a virtual queue has been used many times, for example, in an admission control context [19].

Properties of DSD

Let us list some of the most important properties of DSD:

- Buffer space constraint: The total buffer occupancy for real packets (green and blue counted together) is always less than *Buff*, the size of the virtual queue used for duplicates.
- All accepted blue packets will be served by their deadlines. Accepted blues are thus served at the same time as, or earlier than, they would have been in flat best effort.
- All green packets are served before d, or otherwise dropped. Low bounded (per hop) delay for the green packets is enforced by dropping a green packet that waits or would have to wait d seconds in the queue.
- The green acceptance test does not unnecessarily drop green packets, in the following sense. If all enqueued green packets are to be served, then it is impossible to serve, within d seconds, an incoming green packet that arrived at time t and would violate the green admission test. In addition, if $g = 1$, the green admission test is optimal in the sense that it accepts exactly the green packets that will be served within d s; otherwise not. Note that if $g < 1$, some green packets may become stale and be dropped by the packet serving algorithm.

Points 2 and 3 follow immediately from an earlier section and the pseudo-code. Points 1 and 3 are proven in [20].

Control Loop for DSD

For the reasons described earlier, unlike local transparency, maintaining throughput transparency is by its nature approximate. g is used as a control parameter to balance the throughputs of green and blue. These are estimated from Eq. 1, using a fixed value to represent the nonqueuing delay portion of the round-trip time of a flow. This value is chosen to be small, since this favors blue traffic. For purposes of control, flows are assumed to be greedy, since this also increases the protection to blue flows.

Estimates of the delay and loss ratio for both green and blue traffic are monitored. Let $\theta_b(t)$ and $\theta_g(t)$ be these estimates for the blue and green throughput, respectively, at time t. The value of g is chosen so that their ratio is close to a desired value γ, which is slightly larger than 1, to provide blues with a small advantage in throughput, and to offer a safety margin for protection from errors in throughput estimation.

g is updated every T s according to the control law,

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

where $\alpha \in (0,1)$ and $K > 0$ are two control parameters. T is a chosen parameter of the system which determines the rate of update of g. The initial value of g upon commencement of control can be chosen to be 1, namely, $g(0)1$.

Let us briefly explain the rationale behind this choice of control law, which we do not claim to be optimal. In the ideal case where $\theta_b = \gamma\theta_g$, there should not (a priori) be any bias against blue or green, and the value of g should be 1/2. If θ_b is larger than $\gamma\theta_g$, g must be increased, and vice versa if θ_b is smaller than $\gamma\theta_g$. We wish to maintain symmetry in the amount by which we increase or reduce g: the amount by which g is increased if $\theta_b/\gamma\theta_g$ is multiplied by some factor A should be the same amount by which g is decreased if $\theta_b/\gamma\theta_g$ is divided by the same factor A. Denoting by $\xi = \ln(\theta_b/\gamma\theta_g)$, the targeted g should therefore be an increasing function F of ξ with central symmetry around 0, and such that $F(0)$ 1/2, $F(\xi)$ 0 for $\xi \to -\infty$ and $F(\xi) = 1$ for $\xi \to +\infty$. Such a function is the sigmoid function

$$F(\xi) = \frac{1}{1+\exp(-k\xi)}$$

where K is the slope of the function at the origin. The larger K, the closer the sigmoid function to the step (heavyside) function

Event	What is served?
Both queues empty	Nothing
Green queue empty, blue queue not empty	Head of blue queue
Blue queue empty, green queue not empty	Head of green queue
Head of blue queue cannot wait	Head of blue queue
Head of blue queue can wait, head of green queue cannot	Head of green queue
Heads of green queue and blue queue can wait	With probability g, head of green queue else head of blue queue

Table 1. *Possible events at service time.*

$$F(\xi) = \begin{cases} 1 & \text{if } \xi > 0 \\ 1/2 & \text{if } \xi = 0. \\ 0 & \text{if } \xi < 0 \end{cases}$$

The control law

$$g(t+T) = g(t) + \alpha(F(\xi) - g(t)),$$

where α is the adaptation gain, will therefore bring g to the targeted value. If $0 \le \alpha \le 1$, this control law keeps $g(t)$ between 0 and 1 at all times t. Replacing ξ by $\ln(\theta_b/\gamma\theta_g)$ in this equation, we get the control loop equation for the green bias as given in Eq. 2.

The use and control of the green bias g 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, not just those at the head. Such extensions are the subject of further investigation.

Simulation Results

In this section we show simulations, using ns-2, of DSD run on the topology shown in Fig. 3. There are $n_{b,1}$ blue sources and $n_{g,1}$ green sources with an outgoing link propagation delay of 20 ms (sources of type 1), and $n_{b,2}$ blue and $n_{g,2}$ green sources with an outgoing 10 Mb/s link of propagation delay 50 ms (sources of type 2). All sources pass the 5 Mb/s link L of propagation delay 20 ms, and terminate via a 10 Mb/s link of propagation delay 10 ms. These blue sources are TCP Reno, and the green sources are the TCP-friendly algorithm described in [6]. There is also green traffic which sends a constant rate r (CBR) and passes through the link L.

The router buffer size was 60 packets (i.e., *Buff* = 60), and the maximum delay green can queue, d, was 0.04 s. For simplicity, the size of all packets is fixed at 1000 bytes. The control loop updates its value of g every 0.5 s (i.e., $T = 0.5$), the gain parameter α was 1.1, and the conservative value of 20 ms was taken to be the round-trip time used for estimating throughput. The router distinguishes green and blue by a bit in the packet header. Each simulation ran for 300 s of simulated time.

The first goal of this simulation study was to show that green does not hurt blue, under a variety of conditions; namely, when there are flows of various round-trip times, where green flows may be either TCP-friendly or non-TCP-friendly, greedy or not and where flows may send a mixture of

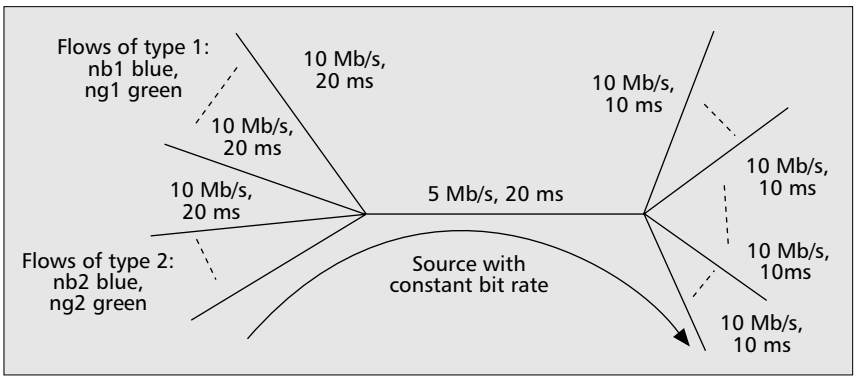
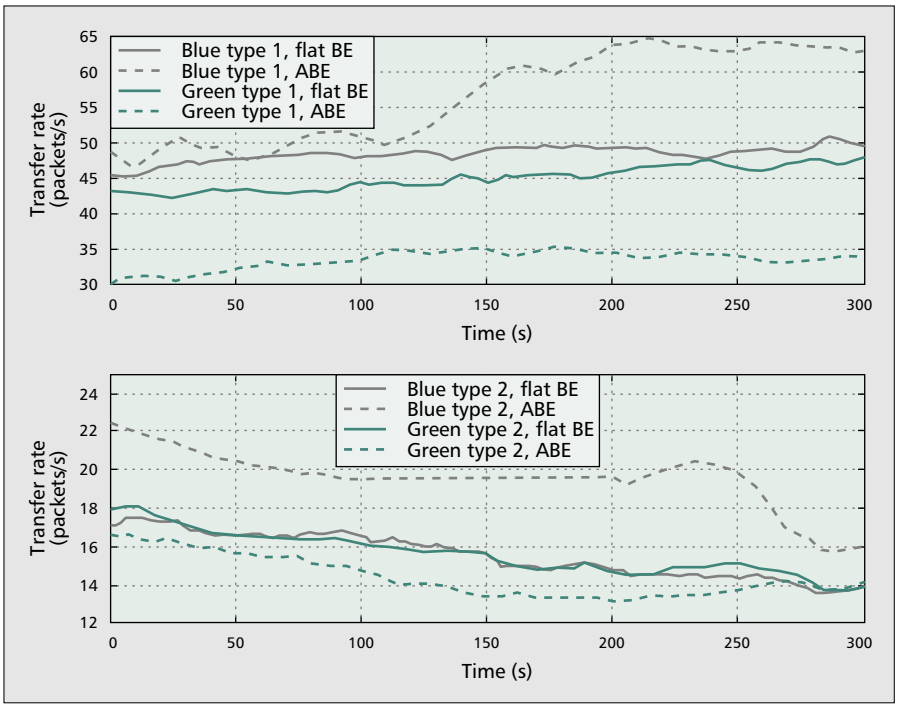


Figure 3. *Simulation topology.*



■ Figure 4. *Average packet transfer rate for green and blue connections, as a function of time* t*, when the router implemented DSD and when it implemented flat best effort. The results are obtained by simulating the network shown in Fig. 3, with 5 blue flows and 5 green flows of each type, namely* $n_{b,1} = n_{b,2} = n_{g,1} = n_{g,25}$*, and no CBR traffic.*

green and blue packets. In addition to this, we illustrate that green flows benefit from low delay (at the expense of less throughput), and the effect DSD has on the loss rates of each traffic type. The reference simulation is what happens in the flat best effort scenario, in the absence of ABE, where all packets are treated equally at the router.

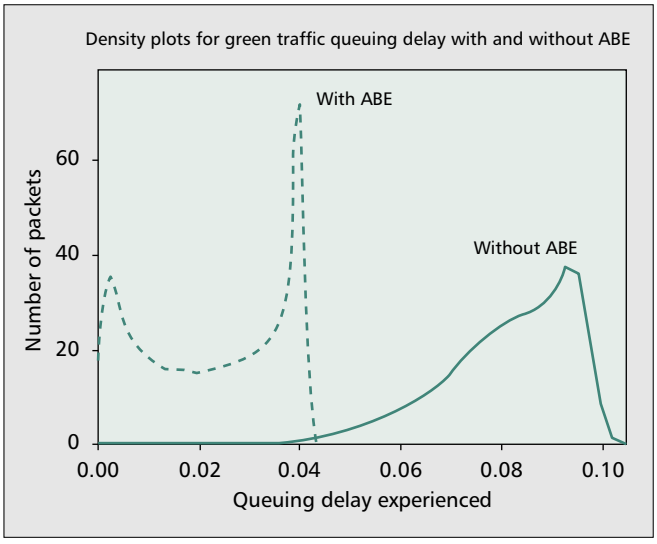
We first examine some scenarios when there are only TCP and TCP-friendly flows. For the case where there are 5 blue TCP and 5 green TCP-friendly flows of each type ($n_{b,1} = n_{b,2} = n_{g,1} = n_{g,2} = 5$), Fig. 4 shows the average transfer rate for each blue and green connection, of both types at each time t. Figure 5 shows the end-to-end delay distributions received for green packets under ABE and flat best effort. Blue flows of each type receive more throughput with ABE than in flat best effort, thus benefiting from the use of ABE. Green flows receive less, and in exchange the green queuing delay is small and bounded by $d = 0.04$ s. The green loss ratio was 4.97 percent when using ABE, and 3.3 percent in flat best effort, while the blue loss ratio decreased from 3.2 to 2.5 percent when moving to ABE. The extra throughput blue flows of type 1 receive over type 2 flows follows from the lower round-trip time they experience.

The same number of blue and green sources does not occur in general, and ABE is designed to work independent of asymmetry in the amount of green and blue traffic. For the case where there are 5 blue TCP and 3 green TCP-friendly flows of type 1 ($n_{b,1} = 5$, $n_{g,1} = 3$), and 3 blue TCP and 5 green TCP-friendly flows of type 2 ($n_{b,2} = 3$, $n_{g,2} = 5$), Fig. 6 shows that again, green does not hurt blue.

The situation where blue traffic is TCP, and green traffic is no longer TCP-friendly, but a constant bit rate (CBR) source, is now examined. Here there are 5 blue TCP flows of each type ($n_{b,1} = n_{b,2} = 5$) and CBR green traffic which sends at 1 Mb/s. The number of packets received for each blue traffic type and for the CBR source is shown as a function of time in Fig. 7. What we see is that the blue traffic receives slightly more throughput with DSD than with flat best effort, due to the local transparency property, and the non-TCP-friendly CBR traffic receives less.

We now look at the scenario where there is blue TCP traffic ($n_{b,1} = n_{b,2} = 5$), and green traffic is composed both of TCP-friendly sources ($n_{g,1} = n_{g,2} = 5$) and CBR traffic of rate 1 Mb/s. The average packet transfer rate for the blue and green of type 1, and the CBR source as a function of time is shown in Fig. 8. The results for type 2 traffic are omitted for ease of reading.

Finally, we look at the case where TCP-friendly flows mix their traffic. In this instance, the flows do not logically decide, based on network conditions, how much traffic to send as blue and how much as green, and this is the subject of further work. Rather, a simple method is used here where, by randomization, the TCP-friendly sources of type 1 send approximately 20 percent blue packets and 80 percent green. Again $n_{b,1} = n_{b,2} = n_{g,1} = n_{g,2} = 5$, and there is CBR green traffic of 1 Mb/s. Figure 9 shows that again, green does not hurt blue. The mixed color sources get a throughput that lies between what they would have gotten were they 100 percent green and what they would have gotten if they were 100 percent blue.



■ Figure 5. *Density plot of queuing delay received by green packet under ABE/DSD and flat best effort. 5 blue TCP and 5 green TCP-friendly flows of each type* ($n_{b,1} = n_{b,2} = n_{g,1} = n_{g,2} = 5$).

Related Work

As the Internet evolves toward a global communication infrastructure, a number of proposals exist for providing QoS architectures. These aim to support more sophisticated services than those provided by flat best effort services. Currently there are two broad families for QoS provision; both are based on some form of priority and service differentiation.

The first family of solutions, integrated services (IntServ), uses reservations (admission control) and requires routers to

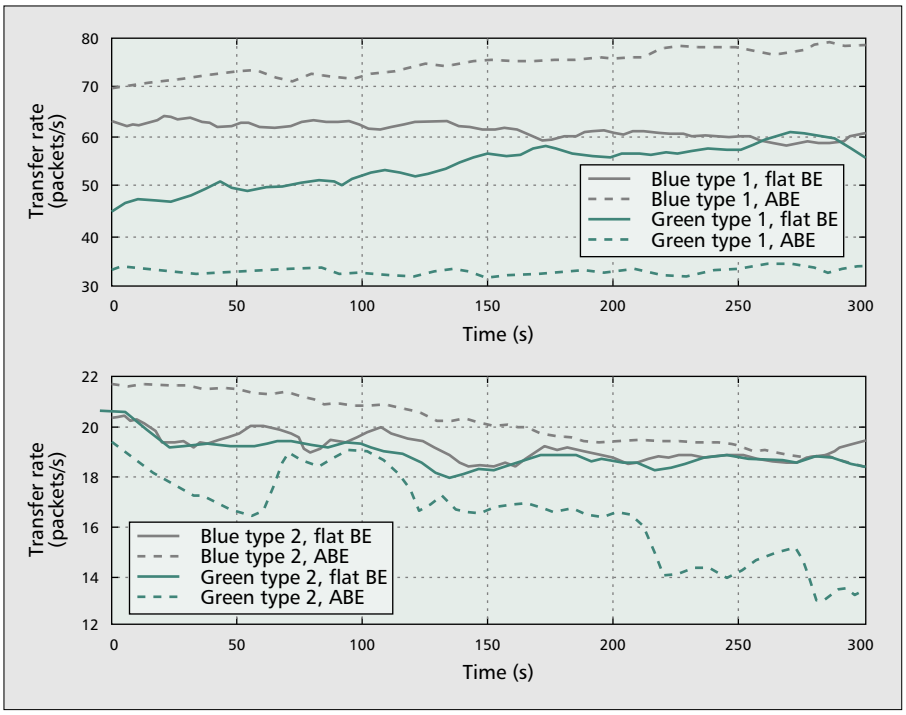


Figure 6. *Average packet transfer rate per green and blue connection, as a function of time t, when the router implemented ABE/DSD and when it implemented flat best effort. The results are obtained by simulating the network shown in Fig. 3, with* $n_{b,1} = 5$, $n_{g,1} = 3$, $n_{b,2} = 3$, $n_{g,2} = 5$.

manage per-flow states and perform per-flow operations. It also requires per-flow accounting and charging. The second family of solutions, differentiated services (DiffServ), is based on a coarser notion of QoS, focusing on aggregates of flows in the core routers and intending to differentiate between service classes rather than provide absolute per-flow QoS measures.

IntServ has been shown to exhibit much higher flexibility and assurance than those provided by DiffServ. However, the main disadvantages of these services are that they are less scalable and robust than DiffServ. Hence, these latter services have been the focus of attention lately mainly because they move the complexity of QoS provision from the core to the edges of the network, where it may be feasible to maintain a restricted amount of per-flow state. Often IntServ are identified as being supported by stateful network architectures (because of the per-flow management), while DiffServ is underpinned by stateless network architectures.

An example of such an architecture is Scalable Core (SCORE) proposed by Stoica and Zhang [21] with the aim of providing guaranteed services without per-flow state management. They proposed the dynamic packet state (DPS) technique to estimate the aggregate reservation rate and use that estimate to perform admission control. To achieve this, they perform core-stateless fair queueing (CSFQ) using DPS to encode dynamic per-flow state in the context of approximating the fair queuing algorithm. Nandagopal *et al.* [22] also proposed a core stateless QoS architecture (called Corelite) which offers per-hop per-class relative average delay differentiation and end-to-end delay adaptation.

There are several proposals for supporting QoS through DiffServ. Crowcroft [23] proposed a low delay service, analyzed by May *et al.* [24], 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 (EF) [25] aims to provide extremely low loss and low queuing delay guarantees. SIMA [26] 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.* [27] described a proportional differentiation model where the quality between classes of traffic is proportional and thus can be performed independent of the load within each class. Central to their work was the utilization of two packet schedulers, backlog proportional rate (BPR) and waiting-time priority (WTP), to approximate the behavior of the proportional differentiation model. Moret and Fdida [28] also described a two-class proportional differentiation model called the proportional queue control mechanism (PQCM). Both studies propose controlling the relative queuing delays between classes.

All of these proposals couple low delay with improved throughput, and use some form of priority. They can be used to support adaptive and nonadaptive interactive applications,

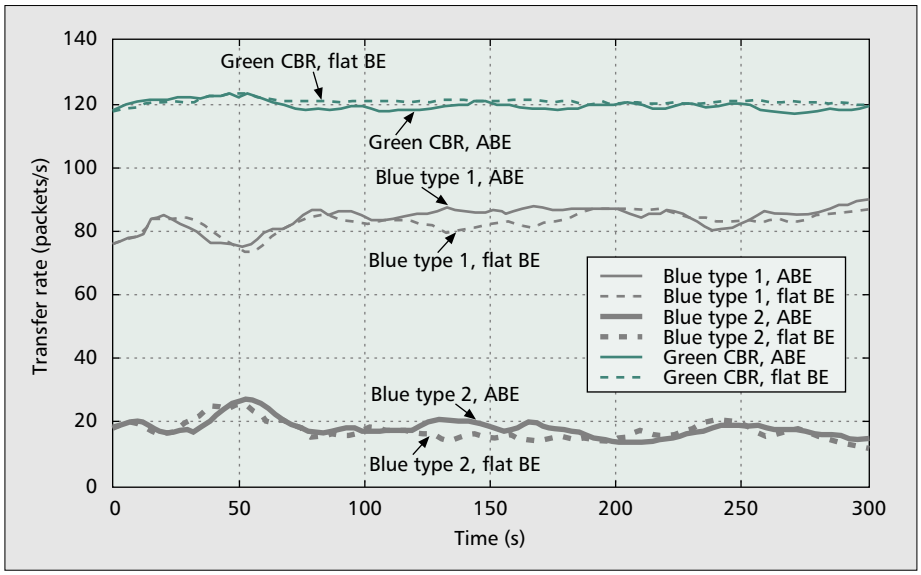
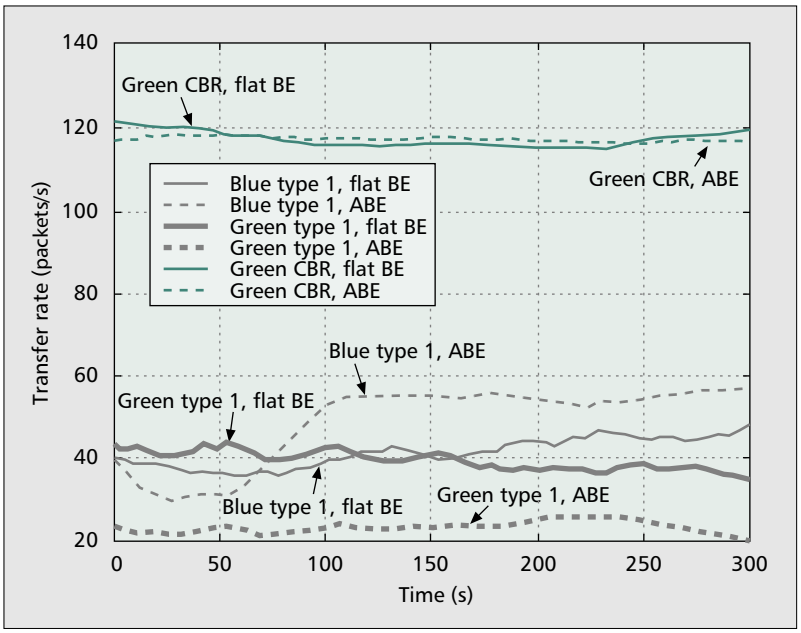


Figure 7. *Average packet transfer rate per green and blue connection, as a function of time* t, *when the router implemented ABE/DSD and when it implemented flat best effort. There are 5 blue flows of each type and a CBR flow of 1 Mb/s which is green.*



■ Figure 8. *Average packet transfer rate per green and blue connection of type 1, and for the CBR source as a function of time* t*, when the router implemented ABE/DSD and when it implemented flat best effort. The CBR source sends at 1 Mb/s, and there are 5 of each other type of source running.*

provided that some form of admission control is performed. They can provide a premium service, at a price that has to be higher than best effort service (otherwise all traffic would use the better service). In contrast, ABE green packets cannot be said to receive 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 multimedia applications which require hard guarantees and cannot adapt.

Assured forwarding (AF) [29] is also a differentiated service. It divides AF traffic into classes within each of which 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. Hence, ABE can be viewed as being positioned between flat best effort service and AF.

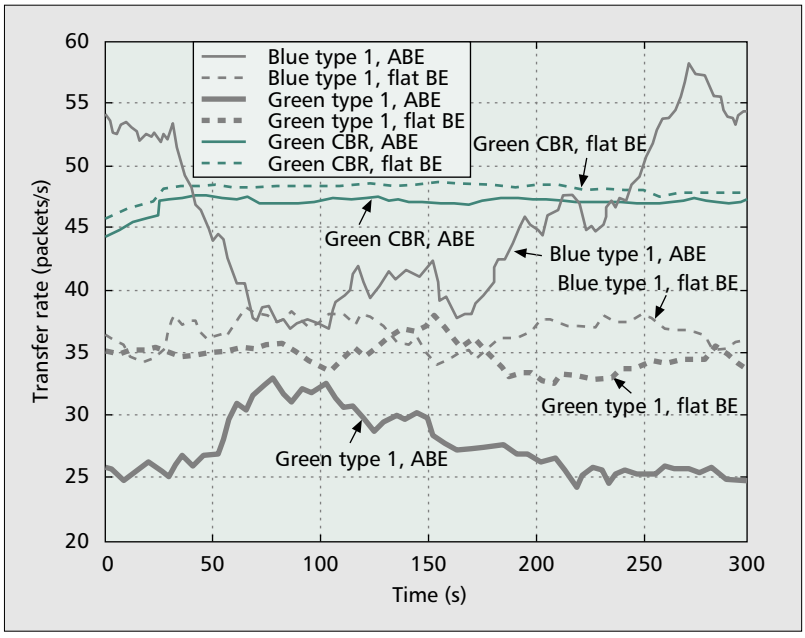
Lastly, destination drop might appear as an alternative to ABE that would require no support from the network. This alternative would consist in having the destination drop all packets that arrive too late, say, after a transit deadline. However, it wastes network resources, since packets are dropped after being carried by the network, and the overall performance of such a scheme can become very poor [14].

Conclusions

We have described ABE, a new service which enables best effort traffic to experience low delay, possibly at the expense of more throughput. ABE is targeted at providing low delay with no concept of reservation or signaling, 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 [6]. Note, however, that ABE also brings benefits if there are nonadaptive UDP applications.

It should be stressed that ABE is a new service in its own right, 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 either pool 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 implementation based on a new scheduling scheme (DSD), and discussed its compliance. Our simulation results show the benefits of the new degree of freedom offered by ABE in best effort services. We have found



■ Figure 9. *Average packet transfer rate 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 shown in Fig. 3, with* $n_{b,1} = n_{g,1} = 3 = n_{b,2} = 3\ n_{g,2} = 5$*, and the TCP-friendly sources of type 1 sent 20 percent blue and 80 percent green packets.*

that under ABE, blue packets received more throughput than under a flat best effort network while giving a low bounded delay to green packets.

References

[1] J. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-Based Error Control for Interactive Audio in the Internet." *Proc. IEEE INFOCOM '99*.
[2] C. Diot, C. Huitema, and T. Turletti, "Multimedia Application Should Be Adaptive," *HPCS*, Aug. 1995.
[3] C. Huitema, "Quality Today in the Internet," ftp.telecordia.com/pub/huitema/stats/quality.today.html
[4] ns v2 simulator. http://www.isi.edu/nsnam/ns
[5] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Comp. Commun. Rev.*, vol. 24, no. 5, Oct. 1994, pp. 10–23.
[6] C. Boutremans and J. Y. Le Boudec, "Adaptive Delay Aware Error Control for internet Telephony," Tech. rep. DSC/2000/031, EPFL-DSC, http://dscwww.epfl.ch, 2000.
[7] B. Suter *et al.*, "Design Considerations for Supporting TCP with Per-Flow Queueing," *Proc. IEEE INFOCOM '98*.
[8] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Net.*, Aug. 1999.
[9] TCP Friendly Web site, http://www.psc.edu/networking/tcp friendly.html
[10] J. Padhye *et al.*, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *Proc. SIGCOMM '98*.
[11] M. Vojnovic, J.-Y. Le Boudec, and C. Boutremans, "Global Fairness of Additive-Increase and Multiplicative-Decrease with Heterogeneous Round-Trip Times," *Proc. IEEE INFOCOM '2000*, Tel Aviv, Israel, Mar. 2000.
[12] T. Henderson *et al.*, "Improving Fairness of TCP Congestion Avoidance," *Proc. IEEE GLOBECOM '98*, Sydney, Australia, Nov. 1998.
[13] "The Alternative Best-Effort Service," Internet draft, draft-hurley-alternative-best-effort-01.txt, work in progress.
[14] P. Hurley, J. Y. Le Boudec, and P. Thiran, "The Alternative Best-Effort Service," Tech. rep./res. rep. DSC1999/036, EPFL-DSC, http://dscwww.epfl.ch, 1999.
[15] P. Hurley and J. Y. Le Boudec, "A Proposal for an Asymmetric Best-Effort Service," *Proc. IEEE/IFIP IWQoS '99*, London, England, May 1999
[16] ABE Project Web page: http://www.abeservice.org
[17] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Net.*, vol. 1, no. 4, Aug. 1993, pp. 397–413.
[18] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proc. IEEE*, vol. 83, no. 10, Oct. 1995.
[19] T. Ferrari, W. Almesberger, and J. Y. Le Boudec, "SRP: A Scalable Resource Reservation Protocol for the Internet," *Comp. Commun.*, Sept. 1998, vol. 21, no. 14, Special Issue on Multimedia Networking, pp. 1200–11.
[20] P. Hurley *et al.*, "A Novel Scheduler for a Low Delay Service Within Best-Effort," *Proc. IEEE/IFIP IwQoS 2001*, Karsruhe, Germany.
[21] I. Stoica and H. Zhang, "Providing Guaranteed Services without Per Flow Management, *Proc. ACM SIGCOMM '99*, pp. 81–94.
[22] T. Nandagopal *et al.*, "Relative Delay Differentiation and Delay Class Adaptation in Core-Stateless Networks," *IEEE INFOCOM 2000*, Tel Aviv, Israel.
[23] J. Crowcroft, "All You Need is Just 1 bit," Keynote Presentation, *IFIP Conf. Protocols for High Speed Networks*, Oct. 1996.
[24] M. May *et al.*, "1-bit Schemes for Service Discrimination in the Internet: Analysis and Evaluation," Tech. rep. no. 3238, INRIA.
[25] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," RFC 2598.
[26] K. Kilkki, and J. Ruutu, "Simple Integrated Media Access — An Internet Service Based on Priorities," *6th Int'l Conf. Telecommun. Sys.*, 1998.
[27] C. Dovrolis, D. Stiliadia, and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," *Proc. ACM SIGCOMM '99*.
[28] Y. Moret and S. Fdida, "A Proportional Queue Control Mechanism to Provide Differentiated Services," *Int'l. Symp. Comp. Sys.*, Belek, Turkey, Oct. 1998.
[29] J. Heinanen *et al.*, "Assured Forwarding PHB Group," RFC 2597.

Biographies

PAUL HURLEY (paul.hurley@epfl.ch) is a research assistant at EPFL, where he will shortly finish his Ph.D. thesis. He graduated with first class honors in computer and mathematical science from University College, Galway, Ireland, in 1995. He then worked as a design engineer both for Videologic, London, and for Digital (now Compaq) in Galway, Ireland. His main research interest is traffic control in communication networks.

MOURAD KARA is a lecturer and head of the networking research group in the School of Computing at the University of Leeds, England. His main research interests are in performance evaluation and engineering of Internet protocols and services to support middleware technologies. He is an advocate of industrial research and has been a research consultant for several companies and industrial laboratories.

JEAN-YVES LE BOUDEC is a full professor at EPFL. He graduated from Ecole Normale Superieure de Saint-Cloud, Paris, France, and received his doctorate in 1984 from the University of Rennes, France, and became an assistant professor at INSA/IRISA, Rennes. In 1987 he joined Bell Northern Research, Ottawa, Canada, as a member of scientific staff in the Network and Product Traffic Design Department. In 1988, he joined the IBM Zurich Research Laboratory at Rueshchlikon, Switzerland, where he was a manager of the Customer Premises Network Department. In 1994 he formed the Laboratoire de Reseaux de Communication at EPFL, which in autumn 1997 became part of ICA. His interests are in the architecture and performance of communication systems.

PATRICK THIRAN received his electrical engineering degree from the Université Catholique de Louvain, Louvain-la-Neuve, Belgium, in 1989, his M.S. degree in electrical engineering from the University of California at Berkeley in 1990, and his Ph.D. from the Swiss Federal Institute of Technology at Lausanne (EPFL) in 1996. He is with the Institute for Computer Communications and Their Applications, where he received the title of professor in 1998. His research interests are in the fields of traffic control in communication networks and system theory. He was on leave from EPFL at Sprint Advanced Technology Labs, Burlingame, California, in 2000–01.