

# A Novel Scheduler For a Low Delay Service Within Best-Effort

Paul Hurley<sup>1 2</sup>, Mourad Kara<sup>3</sup>, Jean-Yves Le Boudec<sup>2</sup>, Patrick Thiran<sup>4</sup>

**Abstract.** We present a novel scheduling algorithm, Duplicate Scheduling with Deadlines (DSD). This algorithm implements the ABE service [5] which allows interactive, adaptive applications, that mark their packets green, to receive a low bounded delay at the expense of maybe less throughput. ABE retains the best-effort context by protecting flows that value higher throughput more than low bounded delay, whose packets are marked blue. DSD optimises green traffic performance while satisfying the constraint that blue traffic must not be adversely affected. Using a virtual queue, deadlines are assigned to packets upon arrival, and green and blue packets are queued separately. At service time, the deadlines of the packets at the head of the blue and green queues are used to determine which one to serve next. It supports any mixture of TCP, TCP Friendly and non TCP Friendly traffic. We motivate, describe and provide an analysis of DSD, and show simulation results.

## 1 Introduction

We describe and analyse Duplicate Scheduling with Deadlines (DSD), a novel scheduling algorithm which implements ABE [5], an enhancement to the IP best-effort service to provide low queuing delay at the expense of maybe less throughput. In order to understand the reasoning behind and advantages of DSD, we provide a brief overview of ABE. A full description and discussion of ABE can be found in [5]. ABE does not need to police how much traffic opts to use low delay, and retains the operational simplicity of a single class best-effort network. ABE packets are marked either green or blue, with green packets receiving a low, bounded delay at every hop. For the service to remain best-effort with no overall advantage to either traffic type, sources which choose not to avail of the lower delay must receive at least as good a service as they would as if all packets had been blue. The introduction of ABE must be transparent to them.

As such, ABE requires that *green does not hurt blue*. If some source decides to mark some of its packets green rather than blue, then the quality of service received by sources that mark all their packets blue must remain the same or become better. The delay and throughput of blue sources must not deteriorate and be at least as good as it was if a “flat best-effort” network, namely if all

---

<sup>1</sup> Part of this work was done while working at Sprint ATL. Contact author. paul.hurley@epfl.ch, Ph. +41-21-693-6626

<sup>2</sup> Institute for Computer Communication and Applications (ICA), EPFL, Switzerland

<sup>3</sup> School of Computing, University of Leeds, United Kingdom

<sup>4</sup> Sprint ATL, Burlingame, CA, USA

packets were blue. As a consequence, green packets are more likely to be dropped during periods of congestion than blue ones. This requirement applies whether green traffic originates from TCP Friendly sources, those which receive no more throughput than a TCP flow would, or from non TCP Friendly flows. Despite the mandate that non TCP sources be TCP Friendly [3], it is still the case that many multimedia flows are not TCP Friendly. It is worth mentioning that, with or without ABE, non TCP Friendly sources may, in some cases, severely hurt other, TCP Friendly sources. The ABE requirement simply means that giving low delay to such sources does not make things worse.

The overall design goal of DSD is to provide green with the best possible service while still ensuring protection for blue, namely that green does not hurt blue. Any significant extra gain by blue packets would be at the expense of green ones, reducing the incentive to choose green. In ABE, the first part of the green does hurt blue requirement is *local transparency*, which is satisfied, if, for each blue packet in the ABE scenario:

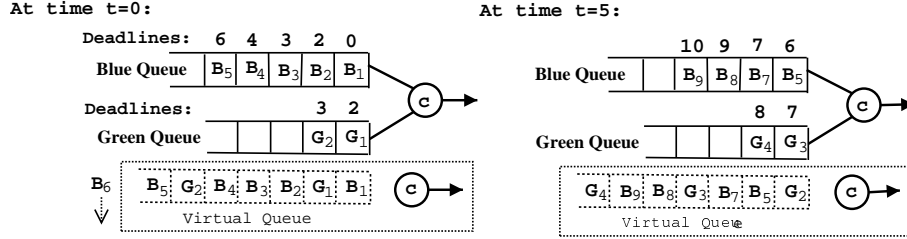
1. the delay is not larger than it would have been in a flat best-effort scenario, where a node would treat all ABE packets as one single best effort class;
2. it is dropped only if it would have been in the flat best-effort scenario.

DSD is a solution to the optimisation problem, which minimises the number of green losses subject to the following constraints:

- Green packets receive a no larger queueing delay than  $d$  (thus satisfying the low delay requirement).
- Local transparency to blue holds.
- The scheduling is work conserving.
- No reordering: Blue (respectively green) packets are served in the order of arrival.

DSD sends a duplicate of each packet arrival to a virtual queue. A blue packet is only dropped if its duplicate was in the virtual queue. Otherwise it receives a deadline equivalent to the service time its duplicate has in the virtual queue. A green packet is accepted if it passes what is called the *green acceptance test*, and then assigned a deadline equal to its arrival time plus the maximum time it can wait in the queue  $d$ . 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 one is to be served next; blue packets served at the latest their deadline permits and green served in the meantime if they have been in the queue for less than  $d$  seconds, and are dropped otherwise.

The virtual queue is not restricted to drop-tail queueing (although in the simulation results we show it is). An Active Queue Management scheme such as RED [7] can be supported for blue traffic by applying it to the virtual queue, and using those results in assigning losses and deadlines. 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)[8] schedulers and its variants. However, EDF sorts packets according to deadlines, whereas DSD remains FIFO within each of its two queues, and the deadlines are used at service to determine whether the head of the green



**Fig. 1.** Two snapshots as an example of DSD, at time  $t = 0$  (left) and  $t = 5$  (right). 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  $G_1$ ,  $B_2$ ,  $B_3$ ,  $B_4$ ,  $G_2$  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  $B_9$

or the head of the blue queue should be served. The use of a virtual queue has been used many times, for example in an admission control context [9].

The second part of green does not hurt blue is throughput transparency. If some sources sending green traffic are rate-adaptive (TCP Friendly), and greedy, local transparency to blue may not be sufficient. It is quite possible that, by becoming green, a TCP Friendly source would achieve a higher data rate, due to the reduction in round-trip time (which is a direct result of the known bias in TCP in favour of flows with shorter round-trip times). To provide throughput transparency, an ABE node must 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, since it requires knowledge of the round-trip time for every flow, which is not feasible in practice and the rate adaptation algorithm implemented by a source may significantly deviate from strict TCP friendliness. Indeed, the dependency of rate on round-trip time is not necessarily a desirable feature of a rate adaptation algorithm.

To provide throughput transparency in the DSD scheduler, a controller, as described in Section 4, acts upon a parameter  $g$  to control the service received by green packets.  $g$  is the probability of serving the green queue first 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, when there is a tie so to speak. The delay and loss ratio are monitored, and the controller adjusts  $g$  to make sure throughput transparency is maintained.

In the next section, a full description for DSD is given, followed in Section 3 by a presentation of some of its properties. A control loop for DSD is described in Section 4, followed by simulations of DSD in Section 5.

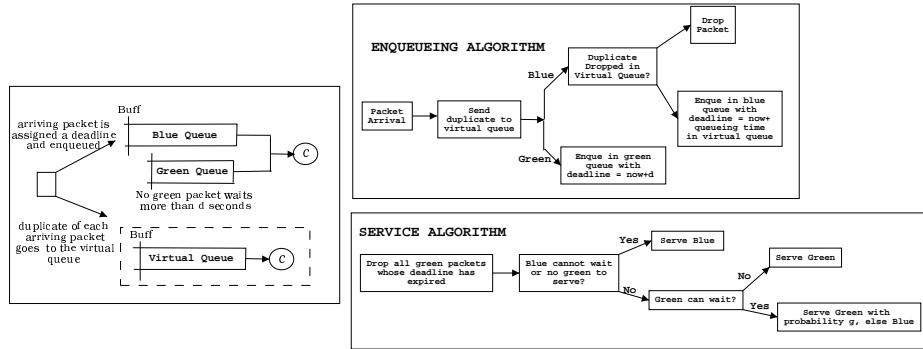


Fig. 2. Outline of the DSD algorithm

## 2 Scheduler Description

An example of how DSD works is given in Figure 1, while Figure 2 provides an overview of the algorithm. It is assumed the router has only output port queueing. Duplicates of all incoming packets are sent to a virtual queue with a 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 a flat best-effort. The times at which duplicates will be served are used to assign blue packets *deadlines* at which they would have been served in flat best-effort. The original arriving packets are fed according to their colour 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  $d$  seconds, and are 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  $t + d + pg_{new}$ , where  $pg_{new}$  is the transmission delay for the incoming green packet, is more than  $c(d + pg_{new})$ , 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 in Section 3. Consider again the example in Figure 1, 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 Figure 1 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 was 5 packets (including  $G_2$ ), and 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 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, as 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 favoured 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 favoured. Conversely, the value  $g = 0$  corresponds to the systematic favouring of blue packets. In the example in Figure 1, 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, as equally protected from this type of traffic as they would have been in a flat best-effort service. The value of  $g$  chosen is made according to a control loop which is described in Section 4.

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. Pseudocode for DSD is given in Table 2. 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 optimisations of this algorithm may be needed, and are the subject of on-going work.

### 3 Scheduler Properties

Some important properties of DSD are:

1. 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.
2. 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.

<p style="text-align: center;"><b>Packet Enqueueing Algorithm</b></p> <p>packet <math>p</math> arrives at the output port  <math>dup = p</math>  Add <math>dup</math> to the virtual queue</p> <p>if <math>p</math> is blue  if <math>dup</math> was dropped from virtual queue  drop <math>p</math>  else  <math>vd =</math> queuing delay received by <math>dup</math>  in virtual queue  <math>p.deadline = now + vd</math>  add <math>p</math> to blue queue  else // <math>p</math> is green  if <math>p</math> fails “green acceptance test”  drop <math>p</math>  else  <math>p.deadline = now + d</math>  add <math>p</math> to green queue</p> <hr/> <p style="text-align: center;"><b>“Green Acceptance Test”</b></p> <p><math>pg_{new} =</math> transmission delay for <math>p</math>  <math>l_g =</math> length of green queue  <math>l_b =</math> length of packets in blue queue with  deadlines <math>\leq now + d + pg_{new}</math>  if <math>l_g + l_b &gt; c * (d + pg_{new})</math>  return “<math>p</math> fails test”  else  return “<math>p</math> passes test”</p>	<p style="text-align: center;"><b>Packet Serving Algorithm</b></p> <p>drop stale green packets, those packets from green queue who cannot be served within their deadline</p> <p>headGreen = packet at head of green queue  headBlue = packet at head of blue queue</p> <p>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  <math>p_g =</math> headGreen.transmissionDelay  <math>dead_g =</math> headGreen.deadline  <math>p_b =</math> headBlue.transmissionDelay  <math>dead_b =</math> headBlue.deadline</p> <p>if <math>now &gt; dead_b - p_g</math>  serve headBlue // because it cannot wait  else if <math>now &gt; dead_g - p_b</math>  serve headGreen // because it cannot wait  else with probability <math>g</math>  serve headGreen  else  serve headBlue</p>
---	---

**Table 1.** Pseudocode of DSD.  $now$  is the current time,  $p.deadline$  denotes the latest time a packet  $p$  can remain in the queue (whose value is tagged onto packet  $p$ ), and  $p.transmissionDelay$  denotes its transmission delay.

3. All green packets are served before  $d$ , or are 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.
4. 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. Also, if  $g = 1$ , the green admission test is optimal in the sense that it accepts exactly the green packets that will be served within  $d$  seconds. Note that if  $g < 1$ , some green packets may become stale and be dropped by the packet serving algorithm.

Items 2 and 3 are obvious consequences of the DSD algorithm. Item 1 is Theorem 1, proven in the appendix. Item 4 is Theorem 2, also proven in the appendix.

## 4 Control loop for DSD

For the reasons described in the Introduction (Section 1), 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, which are estimated by the formula,

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

where  $R$  is the round-trip time,  $p$  the loss rate,  $t_1$  the TCP retransmit time (roughly proportional to the round-trip time), and  $s$  the packet size [4].

A fixed value is used to represent the non-queueing delay portion of the round-trip time of a flow. This value is chosen to be small, since this favours blue traffic. For the purposes of the control, flows are assumed to be greedy, since this also increases the protection to blue flows. Estimates for the delay and loss ratio for both green and blue traffic are monitored, and throughput estimated by Equation (1). 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  $\gamma$ , 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  $T$  seconds according to the control law,

$$g(t + T) = (1 - \alpha)g(t) + \frac{\alpha}{1 + (\gamma\theta_g(t)/\theta_b(t))^K}, \quad (2)$$

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 nor green, and the value of  $g$  should be 1/2. If  $\theta_b$  is larger than  $\gamma\theta_g$ , then  $g$  must be increased, and vice versa if  $\theta_b$  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  $\xi$  with central symmetry around 0, and such that  $F(0) = 1/2$ ,  $F(\xi) = 0$  for  $\xi \rightarrow -\infty$  and  $F(\xi) = 1$  for  $\xi \rightarrow +\infty$ . 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 (Heaviside)

$$\text{function } \bar{F}(\xi) = \begin{cases} 1 & \text{if } \xi > 0 \\ 1/2 & \text{if } \xi = 0 \\ 0 & \text{if } \xi < 0. \end{cases} \quad \text{The control law } g(t+T) = g(t) + \alpha(\bar{F}(\xi) - g(t))$$

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

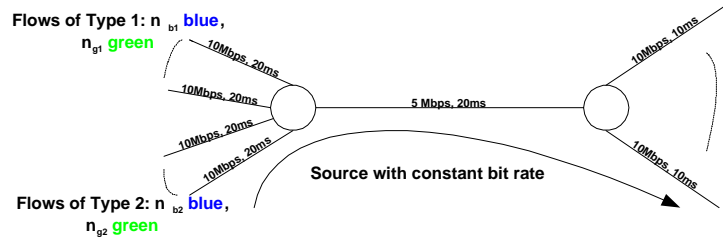


Fig. 3. Simulation topology

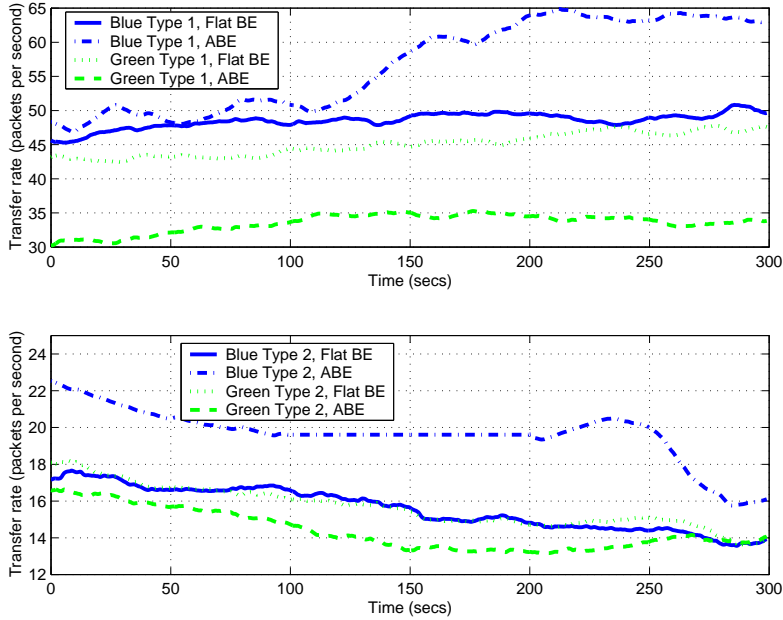
## 5 Simulation of DSD

In this section, we show simulations, using ns-2 [1], of DSD run on the topology shown in Figure 3. There are  $n_{b,1}$  blue sources and  $n_{g,1}$  green sources with an outgoing link propagation delay of 20ms (sources of type 1), and  $n_{b,2}$  blue and  $n_{g,2}$  green sources with an outgoing 10Mbps link of propagation delay 50ms (sources of type 2). All sources pass the 5Mbps link  $L$  of propagation delay 20ms, and terminate via a 10Mbps link of propagation delay 10ms. These blue sources are TCP Reno, and the green sources are the TCP Friendly algorithm as 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 for,  $d$ , was  $0.04s$ . For simplicity, the size of all packets is fixed at 1000 bytes. The control loop updates its value of  $g$  every  $0.5s$  (i.e.  $T = 0.5$ ), the gain parameter  $\alpha$  was 1.1, and the conservative value of  $20ms$  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 seconds 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, may be greedy or not. In addition, we illustrate that green flows benefit from low delay (at the expense of less throughput), and show the effect DSD has on the loss rates of each traffic type. ABE results are compared to the flat best-effort scenario, 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 ( $n_{b,1} = n_{b,2} = n_{g,1} = n_{g,2} = 5$ ), Figure 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 they did in flat best-effort, thus benefiting from the use of ABE. Green flows receive less, and in exchange, the green queueing delay is small and bounded by  $d = 0.04s$ . The green loss ratio was 4.97% when using ABE, and 3.3% in the flat best-effort, while the blue loss ratio decreased from 3.2% to 2.5% when moving



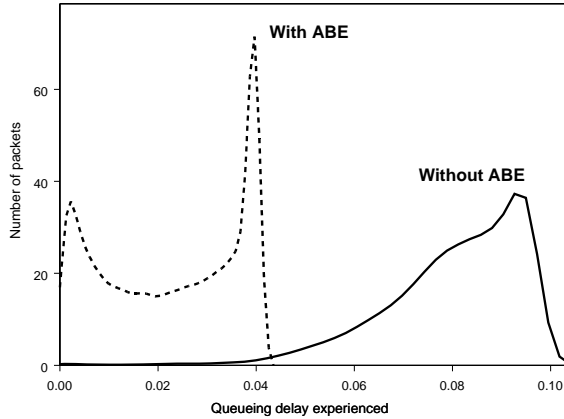
**Fig. 4.** Average packet transfer rate for green and blue connections, as a function of time  $t$ , when the router implemented DSD and when it implemented flat best-effort. The results are obtained by simulating the network described on Figure 3, with 5 blue flows and 5 green flows of each type, namely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

to ABE. The extra throughput that blue flows of type 1 receive over type 2 flows follows from the lower round trip-time they experience.

ABE is designed to work independently of asymmetry in the amount of green and blue traffic. For the case where there are 5 blue 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 ( $n_{b,2} = 3, n_{g,2} = 5$ ), Figure 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 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 1Mbps. The number of packets received for each blue traffic type and for the CBR source is shown as a function of time in Figure 7. What we see is that the blue traffic receives more slightly 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 1Mbps. The average packet transfer rate for the blue and green of type 1, and for the CBR source as a function of time is shown in Figure 8. The results for type 2 traffic is omitted for ease of reading.

Density Plots for Green Traffic Queueing Delay with and without ABE



**Fig. 5.** Density plot of queueing delay received by green packets under ABE/DSD and flat best-effort. 5 blue TCP and 5 green TCP Friendly flows of each type

## 6 Conclusions

We described and analysed a new scheduler, DSD, to enable ABE, a best-effort low-delay service, and thus facilitate multimedia adaptive applications to, in some cases, increase their utility. The freshness in approach involves the assignment of deadlines based on a virtual queue and maximum tolerable queueing delay, and the deadline decision based serving algorithm.

### Appendix: Proof of DSD Properties

We prove items 1 and 4 from Section 3. The notation is illustrated in Figure 9. Denote by  $q_g(t)$  the length of the green queue at time  $t$ , by  $q_b(t)$  the total length of the blue queue (regardless of packet deadlines), and by  $q_v(t)$  the length of the virtual queue, at time  $t$  (thus  $q_g(t) = l_g$  in the pseudocode in Table 2).

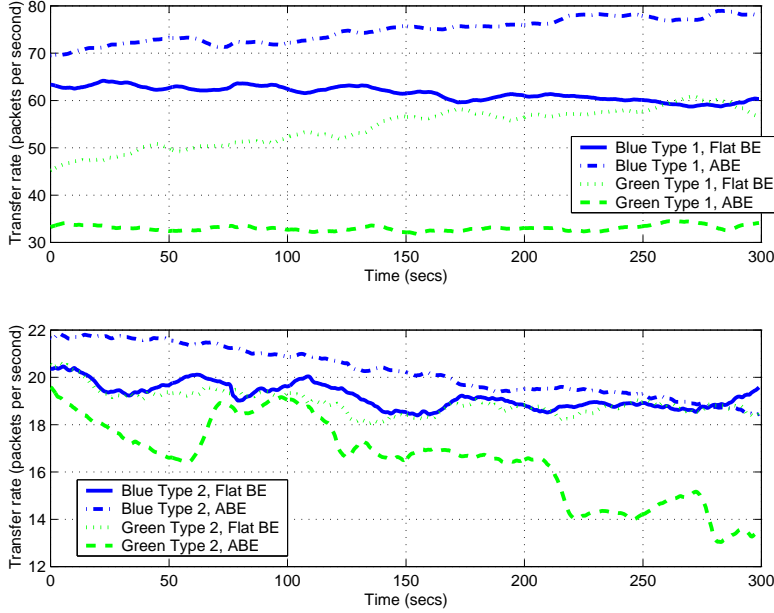
**Theorem 1 (Buffer space constraint).** *The sum of the blue and green queue lengths does not exceed the maximum virtual queue size  $Buff$ : at any time  $t$ ,  $q_b(t) + q_g(t) \leq Buff$ .*

*Proof.* Consider the system at any time  $t \geq 0$ .

(i) Suppose first that  $q_g(t) = 0$ : this means that there are no green packets in the queue at that time. Since a blue packet is accepted if and only if its corresponding duplicate is also admitted, and since it is always served no later than its duplicate, a blue packet will be present in the blue queue if and only if its duplicate is present in the virtual queue. Therefore  $q_b(t) \leq q_v(t)$ . Since the virtual queue length is always less than  $Buff$ ,

$$q_b(t) + q_g(t) = q_b(t) \leq q_v(t) \leq Buff.$$

(ii) Suppose from now on that  $q_g(t) > 0$ . Let  $s$  be the latest time before  $t$  when an incoming green packet has arrived and been admitted, and let  $pg_{new}$  denote its processing time.



**Fig. 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 described on Figure 3, with  $n_{b,1} = 5$ ,  $n_{g,1} = 3$ ,  $n_{b,2} = 3$ ,  $n_{g,2} = 5$

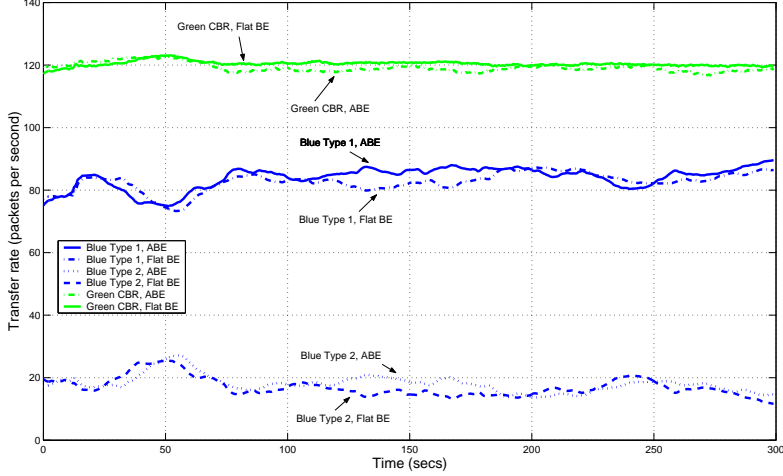
Let  $q_b^{\text{head}}(t, s)$  denote the length of the portion of the blue queue with packets having deadlines in  $[t, s + d + pg_{\text{new}}]$ , (We have thus  $q_b^{\text{head}}(s, s) = l_b$  in the pseudocode in Table 2). It contains the bits that are counted in  $q_b^{\text{head}}(s, s)$ , and that have not been served yet. Likewise, since there are no new green arrivals in  $(s, t]$ ,  $q_g(t)$  contains the bits that are counted in  $q_g(s)$ , and that have not been served yet. Because  $q_g(t) > 0$ , the green queue is never empty in  $[s, t]$ , and therefore the server is never idle during this interval. Therefore

$$q_b^{\text{head}}(t, s) + q_g(t) = q_b^{\text{head}}(s, s) + q_g(s) - c(t - s).$$

At time  $s$ , since the latest green packet passed the green acceptance test,  $q_b^{\text{head}}(s, s) + q_g(s) \leq c(d + pg_{\text{new}})$ . Combining this with the previous equation,

$$q_b^{\text{head}}(t, s) + q_g(t) \leq c(d + pg_{\text{new}}) - c(t - s). \quad (3)$$

On the other hand, let  $q_b^{\text{tail}}(t, s) = q_b(t) - q_b^{\text{head}}(t, s)$  denote the length of the portion of the blue queue at time  $t$  with packets having deadlines larger than  $s + d + pg_{\text{new}}$ . The deadline of an enqueued blue packet is larger than  $s + d + pg_{\text{new}}$  if there are at least  $c(d + pg_{\text{new}} + s - t)$  bits to be served by the virtual server of rate  $c$ , before the corresponding duplicate begins its service. As the total buffer space of the virtual queue is  $Buff$ , the maximum number of bits at time  $t$  in the virtual queue that can belong to duplicates of blue packets with deadline



**Fig. 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 1Mbps which is green

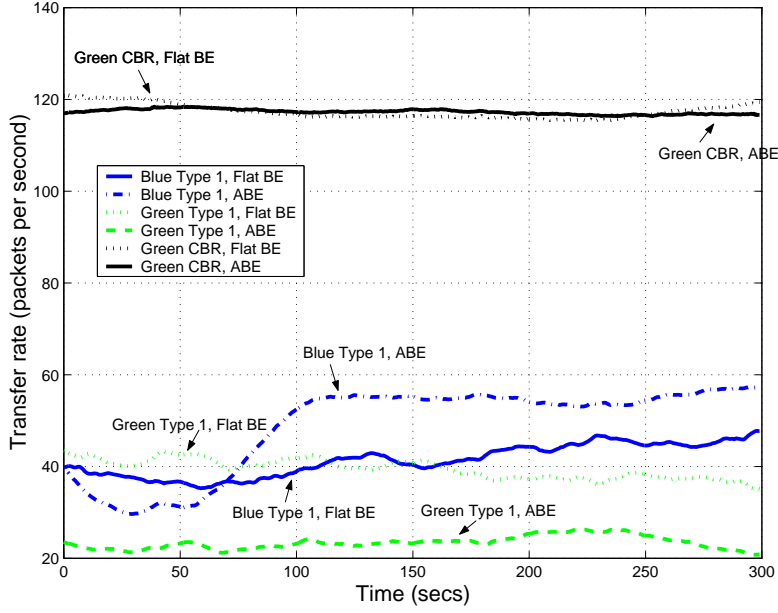
larger than  $s + d + pg_{new}$  is thus at most  $Buff - c(d + pg_{new} + s - t)$ . As a blue packet is present in the blue queue if and only if its duplicate is present in the virtual queue, the length of the portion of the blue queue with packets having a deadline larger than  $s + d + pg_{new}$ , satisfies  $q_b^{\text{tail}}(t, s) \leq Buff - c(d + pg_{new} + s - t)$ . Combining this inequality with (3), we get

$$\begin{aligned} q_b(t) + q_g(t) &= q_b^{\text{tail}}(t, s) + q_b^{\text{head}}(t, s) + q_g(t) \\ &\leq Buff - c(d + pg_{new}) + (s - t) \\ &\quad + c(d + pg_{new}) - c(t - s) = Buff. \end{aligned}$$

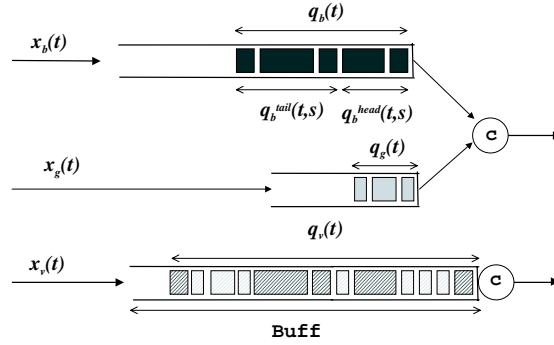
Using the fact that the amount of bits admitted in the virtual queue is the same as the incoming fresh traffic as long as  $q_v < Buff$ , Theorem 1 can be refined and the following lemma established, which will be used in the proof of Theorem 2. The proof of the theorem follows the approach of the proof of Lemma 3, p. 20, in [10]. It states that the sum of green and blue queues at any time is never any larger than the size of the virtual queue.

**Lemma 1 (Virtual Queue Bounds Actual).** *At any time  $t$ ,  $q_b(t) + q_g(t) \leq q_v(t)$ .*

*Proof.* Denote respectively by  $a(t)$  the cumulative amount of traffic (in bits) that has arrived in the router in  $[0, t]$ . Let  $x(t)$  (respectively  $x_v(t)$ ) be the cumulative amount of bits corresponding to packets (resp. duplicates) that have been admitted in the router (resp. in the virtual queue) in  $[0, t]$ . The sum of the backlogs in the blue and green queues at time  $t$  is  $q_g(t) + q_b(t) = \sup_{0 \leq s \leq t} \{x(t) - x(s) - c(t - s)\}$  whereas the virtual queue length at time  $t$  is  $q_v(t) = \sup_{0 \leq s \leq t} \{x_v(t) - x_v(s) - c(t - s)\}$ .



**Fig. 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 1Mbps and there are 5 of each other type of source



**Fig. 9.** Notation used in proofs in Appendix

Let  $t$  be a given time, and let  $0 \leq v \leq t$  be the smallest time such that the virtual queue was not full during the time interval  $[v, t]$ . Because of the duplicates scheme, the traffic that actually entered the virtual system during  $[v, t]$  is therefore identical to the traffic that arrived during this time interval:  $x_v(t) - x_v(v) = a(t) - a(v)$ . If  $v = 0$ , the backlogged data in the actual system

at time  $t$  is given by

$$\begin{aligned} q_g(t) + q_b(t) &= \sup_{0 \leq s \leq t} \{x(t) - x(s) - c(t-s)\} \leq \sup_{0 \leq s \leq t} \{a(t) - a(s) - c(t-s)\} \\ &= \sup_{0 \leq s \leq t} \{x_v(t) - x_v(s) - c(t-s)\} = q_v(t). \end{aligned}$$

If  $v > 0$ , it means that the virtual system was full at time  $v$ , and hence that  $q_v(v) = Buff$ . Then using Theorem 1 to obtain the first inequality below,

$$\begin{aligned} q_g(t) + q_b(t) &= \sup_{0 \leq s \leq t} \{x(t) - x(s) - c(t-s)\} \\ &= \sup_{0 \leq s \leq v} \{x(t) - x(s) - c(t-s)\} \vee \sup_{v \leq s \leq t} \{x(t) - x(s) - c(t-s)\} \\ &= \sup_{0 \leq s \leq v} \{x(t) - x(v) - c(t-v) + x(v) - x(s) - c(v-s)\} \\ &\quad \vee \sup_{v \leq s \leq t} \{x(t) - x(s) - c(t-s)\} \\ &= \{x(t) - x(v) - c(t-v) + \sup_{0 \leq s \leq v} \{x(v) - x(s) - c(v-s)\}\} \\ &\quad \vee \sup_{v \leq s \leq t} \{x(t) - x(s) - c(t-s)\} \\ &= \{x(t) - x(v) - c(t-v) + q_g(v) + q_b(v)\} \vee \sup_{v \leq s \leq t} \{x(t) - x(s) - c(t-s)\} \\ &\leq \{x(t) - x(v) - c(t-v) + Buff\} \vee \sup_{v \leq s \leq t} \{x(t) - x(s) - c(t-s)\} \\ &\leq \{a(t) - a(v) - c(t-v) + Buff\} \vee \sup_{v \leq s \leq t} \{a(t) - a(s) - c(t-s)\} \\ &= \{x_v(t) - x_v(v) - c(t-v) + Buff\} \vee \sup_{v \leq s \leq t} \{x_v(t) - x_v(s) - c(t-s)\} \\ &= \{x_v(t) - x_v(v) - c(t-v) + q_v(v)\} \vee \sup_{v \leq s \leq t} \{x_v(t) - x_v(s) - c(t-s)\} \\ &= \{x_v(t) - x_v(v) - c(t-v) + \sup_{0 \leq s \leq v} \{x_v(v) - x_v(s) - c(v-s)\}\} \\ &\quad \vee \sup_{v \leq s \leq t} \{x_v(t) - x_v(s) - c(t-s)\} \\ &= \sup_{0 \leq s \leq v} \{x_v(t) - x_v(s) - c(t-s)\} \vee \sup_{v \leq s \leq t} \{x_v(t) - x_v(s) - c(t-s)\} \\ &= \sup_{0 \leq s \leq t} \{x_v(t) - x_v(s) - c(t-s)\} = q_v(t). \end{aligned}$$

**Theorem 2.** 1. *If all enqueued green packets are to be served, then the green acceptance test does not unnecessarily drop green packets that could otherwise have been served within  $d$  seconds.*

2. *If  $g = 1$ , then there are no stale green packets.*

*Proof.* Suppose a green packet, with transmission time  $pg_{new}$  arrives at time  $t$ .

(Item 1) Call  $q_g(t-)$  the green queue size just before time  $t$ . We show that if all enqueued green packets in  $q_g(t-)$  are to be served, then it is impossible to serve an incoming green packet that arrived at time  $t$  and would violate the green admission test within  $d$  seconds. To be able to complete the service of the green packet before  $t + d$ , one must be able to serve all packets currently in the green queue, which takes  $q_g(t-)/c$  seconds, all blue packets whose deadline falls

in  $[t, t + d]$ , which takes  $q_b^d(t)/c$  seconds, and the incoming green packet itself, which takes  $pg_{new}$ . The sum of these three times must not exceed  $d$ , which shows that the green acceptance test is indeed necessary.

(Item 2) Suppose  $g = 1$ . Clearly there is enough time to serve all packets currently in the green queue, all blue packets present at time  $t$  and those whose deadline falls in  $[t, t + d]$ , and the incoming green packet itself. Because the queue is FIFO, any green packet that arrives after  $t$  will be served after the green packet that arrived at time  $t$  is served, and hence does not delay the considered green packet which arrived at time  $t$ . One has to check that any accepted blue packet that has arrived after  $t$  does not prevent the green packet that arrived at time  $t$  to be served either. Call  $u$  the smallest time larger than or equal to  $t$  such that  $q_v(u) > c(d + pg_{new})$  (if no such time exists, set  $u = \infty$ ). Then  $q_v(v) \leq c(d + pg_{new})$  for all  $t \leq v < u$ . Because of Lemma 1, the sum of the blue and green queue lengths is less than  $c(d + pg_{new})$ :  $q_g(u) + q_b(u) \leq q_v(u) \leq c(d + pg_{new})$ , which means that all packets that arrived at any time  $t \leq v < u$ , including the green packet that arrived at time  $t$ , will begin their service within  $d$  seconds from their arrival time, in the FIFO order. Conversely, for all  $v \geq u$ ,  $q_v(v) \geq q_v(u) - c(v - u) > c(d + pg_{new}) - c(v - u)$  and hence that any blue packet that arrived at any time  $v \geq u$  will have a deadline to begin its service such that

$$\begin{aligned} v + q_v(v)/c &> v + (d + pg_{new}) - (v - u) \\ &= u + d + pg_{new} \geq t + d + pg_{new}, \end{aligned}$$

i.e. after the green packet under consideration will have completed its service.

## References

1. ns v2 simulator. <http://www.isi.edu/nsnam/ns>
2. S. Floyd, K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, August 1999.
3. TCP Friendly web site. [http://www.psc.edu/networking/tcp\\_friendly.html](http://www.psc.edu/networking/tcp_friendly.html)
4. J. Padhye, V. Firoiu, D. Towsley, J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *Proceedings of SIGCOMM'98*.
5. P. Hurley, M. Kara, J.Y. Le Boudec, P.Thiran. ABE: Providing a Low Delay Service Within Best-Effort. *IEEE Network Magazine*, May 2001. Available at <http://www.abeservice.org>
6. C. Boutremans, J.Y. Le Boudec. Adaptive delay aware error control for internet telephony. Technical Report Research Report DSC 2000/31, EPFL-DSC, <http://dscwww.epfl.ch>, 2000.
7. S. Floyd, V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, V.1 N.4, August 1993, p.397-413.
8. H. Zhang. Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, Vol. 83 No. 10, October 1995.
9. T. Ferrari, W. Almesberger, J. Y. Le Boudec. SRP: a Scalable Resource Reservation Protocol for the Internet. *Computer Communications*, September 1998, Vol 21. No. 14. Special issue on 'Multimedia networking', 1200-1211.
10. J.-Y. Le Boudec, P. Thiran. Network calculus viewed as a Min-plus System Theory applied to Communication Networks. Technical report SSC/1998/016, EPFL, Lausanne, Switzerland, April 1998.