INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2003; **16**:193–210 (DOI: 10.1002/dac.583)

A framework for delivering fine granularity and fair service discrimination in DiffServ networks

Avadora Dumitrescu[†] and Jarmo Harju[*,‡]

Tampere University of Technology, Institute of Communications Engineering, P.O. Box 553, FIN-33101, Tampere, Finland

SUMMARY

In order to achieve a quality of service (QoS) capable of satisfying an ever increasing range of user requirements, differentiated services (DiffServ) have been introduced as a scalable solution that emerges 'naturally' from today's best effort service approach. Mapping the packet treatment into a small number of per hop behaviours (PHBs) is the key idea behind the scalability of DiffServ but this comes at the cost of loosing some behavioural differentiation and some fairness between flows multiplexed into the same aggregated traffic. The paper proposes a novel simple and effective DiffServ approach, the 'Simple Weighted Integration of diFferentiated Traffic' (SWIFT), and uses it in a series of simulations covering a relatively wide range of local network conditions. Measured voice and video traffic traces and computer generated self-similar background traffic were used in simulations performed at various congestion levels and for in-profile and out-of-profile source behaviour. The resulted throughput, mean delay, maximum delay and jitter are used to asses SWIFT's capabilities—isolation of the in-profile traffic from congestion effects, treatment differentiation, increased resource utilization, fairness in treatment under congestion, and incentivity for nice behaviour. Comparisons with other approaches employing traffic control are also provided. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: quality of service; differentiated services; congestion effects; throughput analysis; delay management; jitter control

1. INTRODUCTION

The current Internet is based on the best-effort service model where the network attempts to deliver as many packets as possible without taking into consideration any of the traffic requirements (e.g. delay, delay variation, etc). This approach has been successful until now, when most of the traffic has been, and still is TCP based, relying on the end-to-end congestion mechanism provided by TCP. Yet, plenty of new emerging applications are competing for the Internet resources, and the traditional uniform treatment of all the packets will no longer be

*Correspondence to: Jarmo Harju, Tampere University of Technology, Institute of Communications Engineering, P.O. Box 553, FIN-33101, Tampere, Finland.
† E-mail: avadora.dumitrescu@tut.fi
‡ E-mail: jarmo.harju@tut.fi

Contract/grant sponsor: Graduate School in Electronics, Telecommunications and Automations (GETA)
Contract/grant sponsor: National Technology Agency of Finland (TEKES)

Received 15 September 2002
Accepted 22 November 2002

able to cope with the increasing need for a wide range of explicit requirements. Moreover some 'rogue' applications are trying to grab as much bandwidth as possible by exploiting the back-off property of TCP—opening multiple connections and/or being not responsive in case of congestion—inducing bandwidth starvation for the low bandwidth and 'TCP-friendly' sources. Such situations raised the fair resource allocation to a very important role in congestion control.

To address the multitude of QoS issues, the Internet Engineering Task Force (IETF) has proposed two different service models, namely Integrated Services (IntServ) [1] and Differentiated Services (DiffServ) [2]. IntServ is inherently a reservation based architecture, which provides per flow end-to-end QoS service on condition that *all* the routers along an end-to-end network path understand the reservation protocol. IntServ implies that each router has to manage per-flow state and perform per-flow processing, leading to a very high processing overload in the core routers, a situation that makes IntServ deployment very difficult or even impossible on a large scale.

The DiffServ approach aims to provide a natural evolution path from the current best effort service model, capable to provide service discrimination among *traffic aggregates* over a long timescale while preserving the salient feature—scalability, by pushing the complexity at the edge of the network and keeping the core nodes simple. Inside a DiffServ domain, a packet is processed based on a small number of service levels or *per hop behaviours* (PHBs). Mapping the flow's behaviour into a small number of behaviour classes clearly improves scalability but it comes at the cost of loosing granularity and a great deal of fairness in end-to-end service delivery. Several studies, as those reported in References [3, 4], analyse the effect of traffic aggregation on the individual flow's conformance, identifying sets of cases and combinations of traffic parameters where significant deviations between individual requirements and aggregate performance can exist. Consequently, the limited number of PHBs may induce similar treatment for flows with significant requirement and behaviour differences.

In this paper, we propose a simple and effective DiffServ model, the 'Simple Weighted Integration of diFferentiated Traffic' (SWIFT) protocol, capable of delivering scalable services while preserving a fine granularity in service discrimination. The second chapter of the paper presents the SWIFT algorithm description and points out its particular features. The third chapter gives simulation results using voice, video and best effort traffic under different congestion levels and for sources with various behaviours, and analyses the traffic throughput, the mean and maximum delay and the jitter when SWIFT is employed. Section 4 of the third chapter provides comparison with some other mechanisms employing congestion control features. In the end we summarize the capabilities exhibited by SWIFT and outline future work.

2. SWIFT-ALGORITHM DESCRIPTION

The main idea behind DiffServ in general and our approach in particular is that no per flow state is maintained and no signaling between interior nodes is required, the differentiated treatment of the traffic being derived from information carried by the packets themselves. Along this line we use the *dynamic packet state* (DSP) concept [5–7] in which each packet carries in its header some encoded information (state) that is initialized by the ingress router. The packet is processed (forwarded or dropped) based only on this information carried by its header and on the router's internal state.

The SWIFT algorithm has several components: congestion state determination, packet admission decision, packet buffering and packet information updating (re-labelling). The SWIFT information carried by each packet includes the maximum remaining delay (*max_rd*), when the packets have delay restrictions, and the ratio between the bandwidth reservation (*rsv*) and the average bit rate (*abr*) for the flow to which the packet belongs. The initial values of *max_rd* and *abr/rsv* are set at the ingress node—the *max_rd* and *rsv* being determined according to the flow's service level agreement (SLA), while *abr* is measured, for example using an exponential moving average (EMA) method. Subsequently the values for *max_rd* and *abr/rsv* are actualized after each hop. In case that the packet has no delay restrictions the maximum remaining delay (*max_rd*) is set to zero upon entering the ingress node and when the packet belongs to best effort traffic both the *abr/rsv* ratio and the maximum remaining delay values are set to null at the ingress node as there is no bandwidth reservation and no delay requirement. Encoding the ratio between *rsv* and *abr* saves space in the packet header, compared with the case that they would be encoded separately. The quasi-continuous range of traffic information (delay expectation and bandwidth behaviour) that can be carried by packets allows for a fine service granularity, still preserving the scalability of DiffServ approach. Besides the information carried by the packet, the packet treatment also takes into account the router/node's internal congestion state.

2.1. Packet buffering

Since it is possible to build scalable input–output queuing routers that emulate the behaviour of output queuing routers [8] and since the output queuing architecture is easier to analyse and understand, in all our analyses and simulations we considered an output-queuing router where a packet arriving at the input interface is immediately transferred to the corresponding output interface. In order to be able to provide a wide range of forwarding differentiation the nodes are expected to have several buffers, not only with different sizes but also with different forwarding capacity assigned to them—for example like having different weights if weighted round robin (WRR) scheduler is used. In this case, the buffer forwarding capacity (C_i) reflects the proportion of the overall forwarding capacity assigned to the corresponding queue. Although the real rate at which the scheduler drains packets from the queue may differ from the calculated one (e.g. an empty buffer situation), some rate bounds can be guaranteed.

The buffer to which a packet is assigned is decided by comparing the packet's maximum remaining delay divided by the maximum expected number of remaining hops (max_rd/max_nh, i.e. the local delay requirement under the assumption that all remaining hops are equally capable of providing delay margins) with the node buffer delays—calculated as $Buff_iOccup/C_i$. Based on this comparison the packet is assigned to the buffer having the closest delay value smaller than (max_rd/max_nh) and not to the buffer providing minimum delay. If there is no buffer with the expected delay smaller than (max_rd/max_nh) the packet can be assigned either to the buffer with the smallest expected delay—when this smallest expected delay is smaller than *max_rd*, in hope that the excess delay can be handled by the application or compensated on the remaining path—or to the buffer with the largest expected delay—when the smallest expected delay is greater than *max_rd*. The latter alternative would mean that the packet, whose delay requirements cannot be met, will be treated together with packets without delay restrictions. For certain applications, when the delay requirements cannot be met, the packets might be dropped altogether.

2.2. Congestion state determination

To keep track of the congestion an evidence of the buffer occupancy (overall and on each buffer) is maintained. Derived from the buffer occupancy and/or from the node drop rate each node gets a 'colour' marker (with the 'classical' green, yellow and red colours) to describe its congestion state. The 'colour' marker value is determined by comparing the overall packet drop rate and/or the buffer occupancy (overall or maximum) with two pre-established threshold levels: T1 and T2 (a value below T1 yields the 'green' state of congestion, a value between T1 and T2 yields 'yellow' and a value above T2 yields 'red'). The threshold levels are predetermined for each type of comparison in agreement with the network management experience in node congestion. In our simulations the overall buffer occupancy (OBO) was employed.

2.3. Packet admission control

While out of congestion, in 'green' state, all incoming packets are forwarded, under congestion the packet admission decision is based on a comparison between the forwarding capability and the forwarding requirements of the existing traffic. In evaluating the forwarding requirement of the existing traffic at the moment when a certain packet is processed, the EMA-averaged aggregate traffic bandwidth is multiplied (i.e. 'weighted') by the *abr*/*rsv* ratio of the processed packet. In this way the packet admission decision is based on the assumption that the whole traffic is behaving as good or as bad as the flow to which the packet belongs to. Consequently, the comparison, needed for the packet admission decision under congestion, is between C and $(abr/rsv)*Agtf$, where C is the forwarding capacity and $Agtf$ is the (exponential moving) average of the aggregated traffic bandwidth.

Depending on the congestion status the abovementioned comparison is made at different levels. For the 'yellow' congestion state the forwarding capacity and aggregated traffic are considered for the whole node while for the 'red' state they are considered for the buffer with highest congestion. In this way the algorithm ensures that, as long as the reservation does not exceed the capacity, no traffic is blocked even under extreme congestion.

If, under the existing congestion state, the forwarding capacity, C, is bigger than $(abr/rsv)*Agtf$ the packet is accepted. In order to take into consideration the available buffer space and accommodate some level of burstiness and/or random variations of the flow parameters at packet level, the packets are not dropped altogether when C is smaller than $(abr/rsv)*Agf$ but they are still forwarded with a probability:

$$fp = \min\{1, (rsv/abr)*(BuffSize\text{-}BuffOcc\text{-}BuffMarg)/BuffSize\}$$

where *BuffSize* is the buffer size, *BuffOcc* is the buffer occupancy, *BuffMarg* is a buffer margin—all being considered either for the node or just for the highest congested buffer for 'yellow' and 'red' states, respectively. The use of the forwarding probability (fp) implies that the packet may still be forwarded if it belongs to a well behaved flux and/or if there is buffer space available. The probability of dropping a packet increases as its *abr* surpasses the *rsv* and as the occupancy is a greater fraction of the buffer size. Furthermore, the use of *fp* enables a smooth transition between the acceptance and drop decisions and, together with the packet admission decision, enables a selective treatment for packets whose dropping could critically affect the corresponding application. This can be done, without any special signaling, simply by assuring a high *rsv*/*abr* ratio for the 'special' packets. The use of a forwarding probability also explains the need to consider some buffer margin to accommodate the packets being accepted despite the fact that C is smaller than $(abr/rsv)*Agtf$.

2.4. Packet relabeling

The information carried by the packet is initialized at the ingress nodes and is updated at each hop. While the reserved bandwidth (*rsv*) is defined at the ingress node, according to the service level agreement (SLA) and does not change, the encoded ratio between the average bit rate (*abr*) and the reserved bandwidth (*rsv*) is updated in each node according to the packet forwarding probability, which is considered to be one in 'green' state and when C is bigger than (abr/rsv) $*Agtf$. This is because, when assuming a relatively smooth variation in forwarding conditions from one packet to the next of the same flow, the average bit rate of the flow will change according to the forwarding probability:

$$abr\_new = abr\_old \ * fp$$

The remaining delay is also updated after the packet is received by the (next) node. If the actualized remaining maximum delay is negative—i.e. the packet delay has exceeded the maximum acceptable delay set by the application—the packet could either be dropped or forwarded with *max_rd* (and abr/rsv) value(s) set to null, thus indicating it should be treated either as without delay requirements or on a best effort basis.

2.5. Best effort traffic treatment

As there is no reservation for the best effort traffic packets, a zero abr/rsv value is used to identify them. Furthermore, best effort traffic packets have a null *max_rd* since they do not have a delay requirement, either. The SWIFT treatment of best effort traffic follows the principle that all traffic that has a reserved bandwidth, however 'badly' behaving, should be treated preferentially with respect to the best effort traffic. In order to achieve this requirement the forwarding probability function used for best effort traffic is altered by using a sub-unitary correction factor γ and by replacing the rsv/abr factor with the minimum rsv/abr value for the traffic having a reservation:

$$fp(\text{best-effort}) = \min\{\gamma, (\gamma * \min(rsv/abr) * (BuffSize\text{-}BuffOcc)/BuffSize\}.$$

In our implementation the correction factor γ has a given sub-unitary value (for example 0.7) as long as the minimum rsv/abr has a supra-unitary value (i.e. the traffic having a reservation is well-behaved). When the minimum rsv/abr decreases below one (i.e. there is at least one badly behaving flow) the correction factor asymptotically increases to one. This dependence of the correction factor on the sub-unitary rsv/abr was chosen in order to diminish the advantage given to reserved traffic with respect to best effort traffic when the reserved traffic is out-of-profile. The correction function could also be defined to increase to 1 or even above 1 for low sub-unitary rsv/abr in order to discourage extremely malicious behaviour and to eliminate the possibility that a single very badly behaving flow with some reservation deteriorates the treatment of all the best effort traffic.

Also, since the best effort traffic treatment should not be referred to the worst ever behaving traffic with a reservation but rather to the *current* worst behaving traffic with a reservation, the minimum rsv/abr value is continuously updated over a certain time window.

In terms of buffering, as the best effort traffic has no delay requirements (*max_rd* $= 0$), the best effort packets are assigned to the buffer with the largest associated delay, usually the largest buffer, thus accommodating an increased level of burstiness.

3. SIMULATION RESULTS

3.1. Simple experiment

In order to evaluate our algorithm several simulation experiments were conducted. The transfer of simple sources traffic over a single node was first used to illustrate some basic SWIFT features, and then some more complex traffic patterns were transferred over more complicated node configurations in order to observe the characteristics of the algorithm in more realistic circumstances.

In a first simplified scenario three sources, S1, S2 and S3 compete for the capacity of a link both under congestion and without congestion. S1 is a constant bit rate (CBR) source of 4 Mbps with a reserved bandwidth that varies. S2 is an ON-OFF source with a 4 Mbps bit rate in the ON state and a bandwidth reservation of 4 Mbps. Finally S3 is a CBR source of 4 Mbps with no bandwidth reservation (i.e. best effort traffic). The link capacity is 10 Mbps so that the node is not congested during the OFF periods of S2 and significantly congested during the ON period of S2. The T1 and T2 overall buffer occupancy thresholds delimiting the router's internal congestion states (green, yellow, red) were set to $T1 = 0.1$ and $T2 = 0.56$. For buffer size dimensioning we have considered a maximum acceptable per node delay of 75 ms, which corresponds to about 96 KB buffer space for an output link capacity of 10 Mbps. Accordingly three equal buffers of 32 KB each were used.

The drop rate experienced by the three sources, illustrated in Figure 1, is very different from source to source and strongly dependent on congestion. S2, which is a conformant flow, all the time experiences zero drop rate even during congestion proving that 'well-behaved' sources are insulated from congestion effects by SWIFT. S1, which is not well behaved, having a reservation of 2 Mbps for a constant bit rate of 4 Mbps, also experiences zero drop rate because dropping packets from the best effort source S3 regulates the congestion. This shows that sources having a reservation are preferentially treated with respect to best effort traffic even when they are behaving badly.

Figure 2 shows that, when the reservation of S1 drops to zero, thus putting both S1 and S3 into the best effort traffic category, the congestion is solved by fairly distributing the drop rate between S1 and S3. Since the total reservation cannot equal or exceed the capacity, distributing the drop rates among best effort traffic sources ensures that neither of them will be blocked.

Figure 3 shows the overall buffer occupancy variation in time with the bandwidth reservation for S1 as a parameter. The buffer occupancy variation in time is up-shifted with increased reserved bandwidth for S1 as the overall reserved bandwidth is a larger fraction of the existing link capacity. For large reservation values for S1 ($\text{rsv} = 3$ and 4 Mbps), leading to a high buffer occupancy, the OFF period of S2 is no longer enough to empty the buffers. Consequently, at some high level of reservation for S1, the aggregate transmitted bandwidth remains equal to the capacity even during the OFF period of S2 as can be seen in Figure 4.

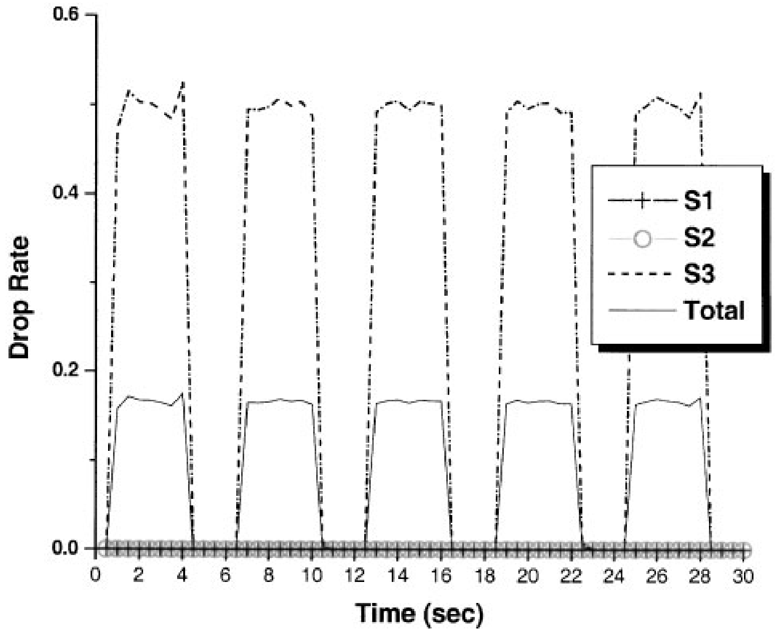

Figure 1. Variation of drop rates in time.

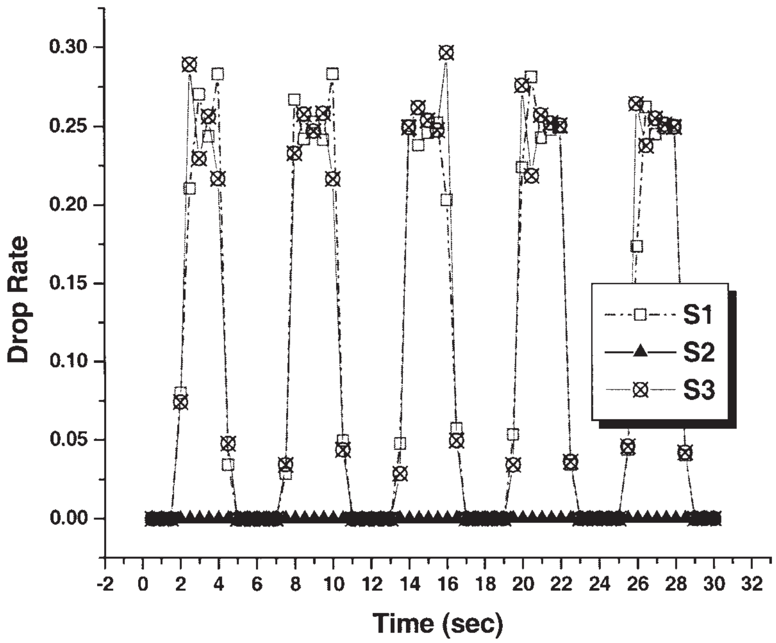

Figure 2. Drop rates for S1 and S3 when they are competing as best effort sources.

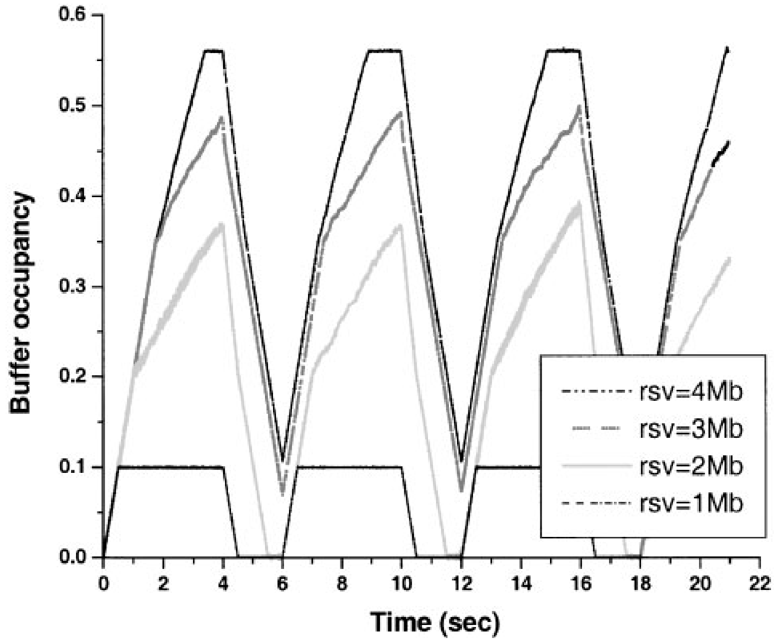

Figure 3. Overall buffer occupancy variation with S1 reservation as a parameter.

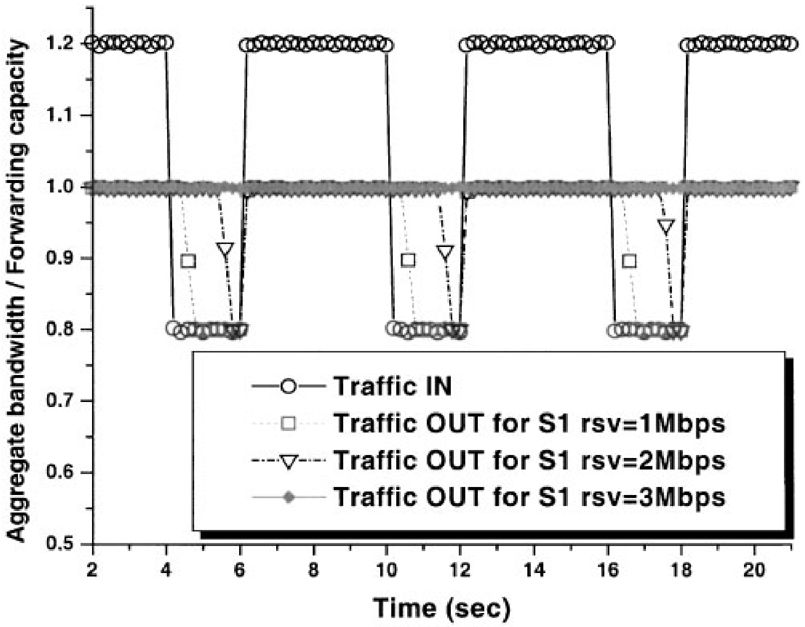

Figure 4. Output aggregate bandwidth variation with S1 reservation as a parameter.

3.2. Comparison with other approaches

In this section the SWIFT algorithm performance is compared with some other existing approaches. We consider the simulation scenario described in Section 3.1 (i.e. three sources—S1, S2, S3—sending traffic through a 10 Mbps link) and the same parameters (user characteristics, buffer sizes, thresholds when applicable) are used for all the simulated approaches tested in comparison with SWIFT:

- FIFO (first in first out)—One of the simplest queuing disciplines without any congestion control mechanism. FIFO employs a drop-tail policy; if the buffer is full, all the incoming packets are discarded.
- RED (random early detection)—Still preserves the first-come first-served queuing discipline but it has an additional active buffer management congestion control mechanism. RED [9] signals congestion by randomly dropping packets once the queue size is above a minimum threshold (*minth*), causing the responsive sources (e.g. TCP) to back-off their transmission rates. If the exponentially averaged queue size becomes larger than the second threshold (*maxth*), all the packets are discarded. When the averaged buffer occupancy is between the two thresholds, the probability of dropping linearly increases with the queue occupancy up to some maximum value (*Pmax*). Another parameter considered in RED is a weighting factor (*wq*) that influences the calculation of the average queue size.
- RIO (random early detection with in and out)—Based on RED mechanism, RIO [10] introduces an additional complexity in buffer management by discriminating between packets belonging to sources whose traffic characteristics are within agreed traffic profile (in-profile) and the packets that are not conforming with the user profile (out-of-profile). The RIO mechanism assumes that packets have already passed through an upstream meter and marker and they were tagged as 'in' or 'out'. The discrimination between the two types of packets is achieved by separately applying the RED algorithm to 'in' and to 'out' packets with the parameters set in such a way that the policy of dropping the 'out' packets is more aggressive than for the 'in' packets (with lower thresholds and higher dropping probability).

The simulation settings used for RED were $Pmax = 0.02$, $wq = 0.002$ [9], $minth = 0.1$ and $maxth = 0.6$, while for RIO $min\_in = 0.4$, $max\_in = 0.6$, $Pmax\_in = 0.02$, $min\_out = 0.1$, $max\_out = 0.3$ and $Pmax\_out = 0.05$ were used. For evaluating the influence of the source behaviour the *rsv* for S1 was varied from 0 to 4 Mbps with a 1 Mbps step. The same variation was used for the token rate used in RIO.

Figure 5 shows the throughput experienced by S1 when FIFO, RED, RIO and SWIFT were used. The throughput obtained by using RIO and SWIFT is given for various reservations made for S1—from $rsv = 0$ to $rsv = 4$ Mbps—while the throughput obtained when using FIFO and RED is shown only once, on the left side of the figure, since in FIFO and RED there is no reservation. It can be observed that the throughput for RIO and SWIFT increases with the reservation and it is always better than the throughput for FIFO and RED. Also it is apparent that SWIFT has a higher throughput with respect to RIO over all the reservation range.

The same kind of analysis carried out for the throughput of S2 shows that the throughput when using RIO and SWIFT is again higher than that for FIFO and RED, being almost 100% and practically constant with the variation of S1 reservation. This indicates that in-profile sources are insulated from congestion consequences as long as the congestion can be solved at the expense of best effort and out-of-profile sources.

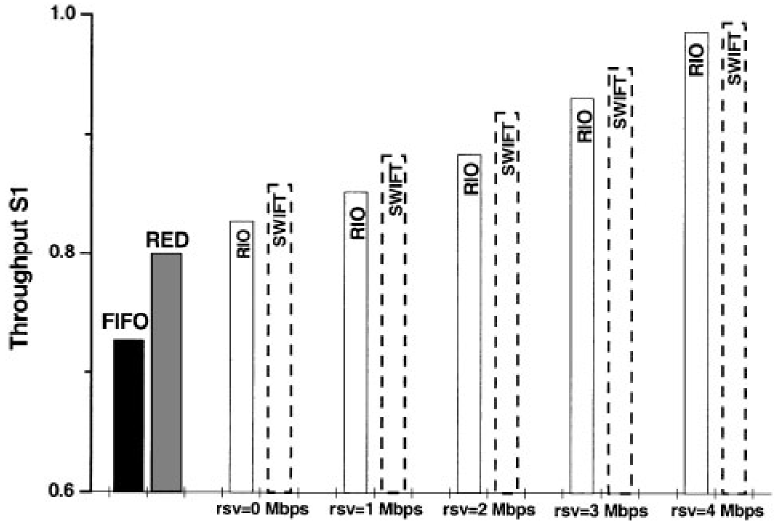

Figure 5. Throughput for S1 when using FIFO, RED, RIO and SWIFT—for RIO and SWIFT the throughput is shown for several S1 reservations.

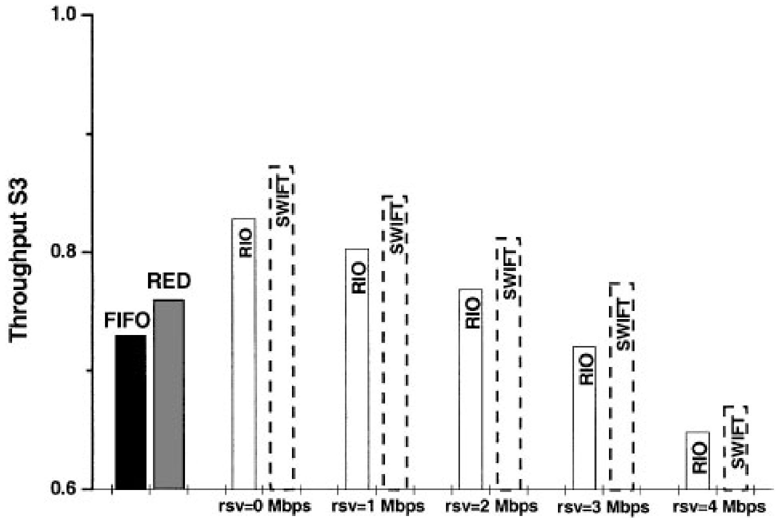

Figure 6. Throughput for S3 when using FIFO, RED, RIO and SWIFT—for RIO and SWIFT the throughput is shown for several S1 reservations.

When looking at the variations of S3 throughput in Figure 6 one notices that the throughput in RIO and SWIFT decreases with increasing S1 reservation as the increased S1 throughput is obtained at the expense of the S3 best effort traffic. The higher throughput of SWIFT with respect to the RIO for S1 and S3 is an indication of the fact that SWIFT accommodates more out-of-profile traffic when resources are available. However, it should be pointed out that, in case of a less optimal choice of parameters, SWIFT could lose its throughput margin with respect to RIO.

We should also make the observation that, in case of fixed timing relations among the sources included in simulation (like in our case) the results can be affected by the systematic relation among the arrival times—a fact that is not so critical when more complex simulation scenarios are used.

3.3. Different traffic source models

In order to assess the performances of SWIFT under more realistic traffic assumptions a scenario with increased traffic burstiness is considered next. We now have 20 Web sources, with an average bandwidth of 125 Kbps, modelled according to References [11, 12]. The sources are ON-OFF with the ON-OFF periods drawn from heavy tailed Weibull and Pareto distribution, respectively. The packet interarrival times corresponding to the ON period follows also a Weibull distribution. The size of the data transfer is drawn from a Pareto distribution with the shaping parameter $k = 1.06$. These sources were considered as best-effort ‘background traffic’. In addition there are four traffic sources (two voice sources and two video sources) for which real-trace measurements, collected directly at packet level using an IP protocol analyzer, were used. The voice traces (S1 and S2) used Speak Freely applications and had average 13 and 32 Kbps bit rates involving GSM and ADPCM coding and compression, respectively. In our simulations S1 had a 15 Kbps reservation and S2 had no reservation. The video traces were obtained from two videoconference sessions and were also collected using the IP analyzer. The trace used for S3, which had a 220 Kbps reservation, was a bursty one with an average bandwidth of 208 Kbps. The trace for S4, for which no reservation was considered, was also quite bursty, with an average bandwidth of 135 Kbps. The link capacity was taken to be just 1 Mbps so that the node is severely congested even when treating just the best effort sources, if several of these sources are simultaneously ON. Under these circumstances dropping best effort traffic solves the congestion and hence S2 and S4 are competing with background www traffic on equal footing.

Figure 7 presents the aggregate throughput of the audio and video sources (S1, S2, S3 and S4) and the aggregated www traffic throughput. It can be observed that the two variations are somehow in ‘antiphase’ due to the ‘opportunistic’ transfer of the www traffic under congestion—i.e. whenever the traffic with a reservation has a burst, thus increasing the aggregate audio and video traffic, the www traffic is cut down and whenever the reserved traffic has a drop in bandwidth the www traffic has the opportunity to increase its throughput. The variations around the average throughput values are induced by the bursts of the reserved traffic, which experiences roughly 100% throughput even when its bandwidth almost covers the forwarding capacity.

Figure 8 shows the throughput of best effort video source (S4) compared with the aggregate throughput of audio and video sources (S1, S2, S3 and S4). It can be seen that the throughput variation of S4 is quite similar with the www traffic throughput variation of Figure 7. This is due not only to the ‘opportunistic’ forwarding of all the best effort traffic but also to the fact that all best effort sources experience a similar treatment. A similar treatment primarily induces a similar average throughput, which can be observed in the pure best effort traces at the bottom of Figures 7 and 8 and can also be deduced from the throughput budget of Figure 7: $13\ \text{Kbps} + 208\ \text{Kbps} + 0.3*(32\ \text{Kbps} + 135\ \text{Kbps}) \approx 0.7*(13\ \text{Kbps} + 208\ \text{Kbps} + 32\ \text{Kbps} + 135\ \text{Kbps})$.

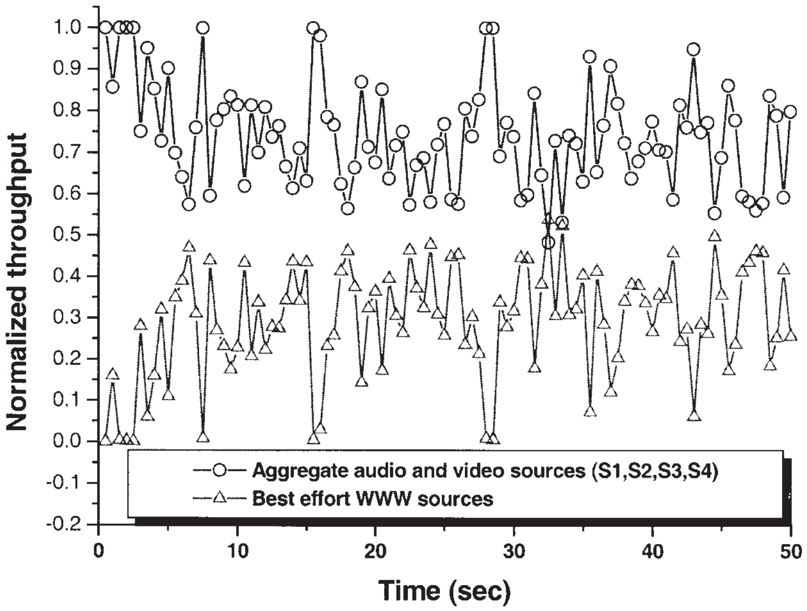

Figure 7. Normalized throughput of the aggregate audio and video traffic compared with that of the aggregate www best-effort traffic.

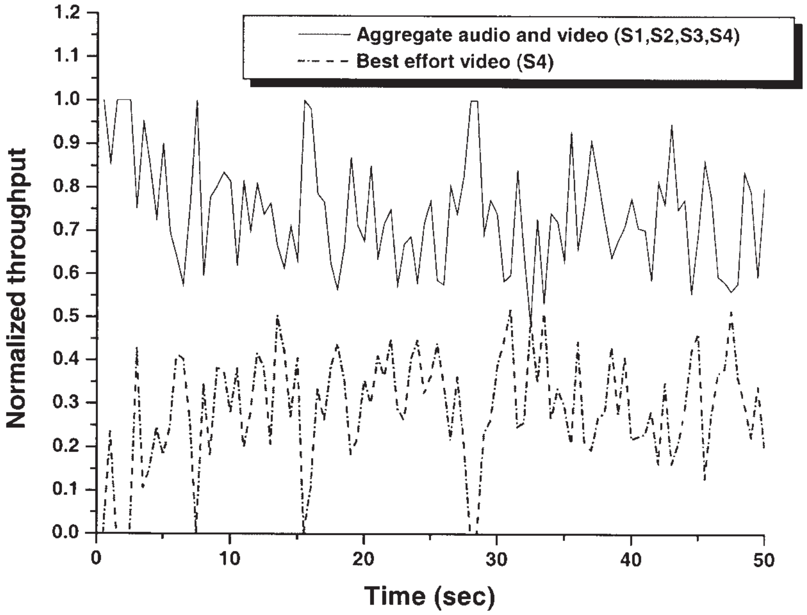

Figure 8. Normalized throughput for the aggregate audio and video traffic compared with that of the best-effort video source (S4).

Table I. Accepted and simulated delay (in ms), over the whole five-node link, for S1, S2, S3 and S4, in different simulation scenarios.

Link delay	S1			S2			S3			S4		
	Acc5	*Mean*	*Max*	Acc9	*Mean*	*Max*	Acc4	*Mean*	*Max*	Acc5	*Mean*	*Max*
Case 1	62.5	2.72	4.08	25.0	2.94	4.31	125.0	4.83	8.55	75.0	4.03	5.56
Case 2	62.5	21.66	42.44	25.0	20.03	28.03	125.0	28.84	80.74	75.0	24.84	70.67
Case 3	18.0	18.23	459.35	15.0	14.84	28.94	80.0	23.97	70.18	65.0	21.32	63.81
Case 4	62.5	27.81	49.34	25.0	21.81	40.58	80.0	28.59	76.19	75.0	27.51	73.96
Case 5	62.5	37.74	61.97	25.0	43.77	1756.94	80.0	50.42	79.94	75.0	69.36	2386.18

Figure 8 also proves that best effort sources are not blocked even under the most adverse conditions, except for the extreme situation (forced in our simulations for exemplification) when the reserved traffic burst covers the whole forwarding capacity.

3.4. *Delay analysis*

In order to evaluate SWIFT's capacity to assure delay requirements we have first simulated a one-way transfer over a congested five-node link path. The traffic with delay restrictions came from the previously described voice sources (S1 and S2—with 13 and 32 Kbps average bit rate, respectively) and two video sources (S3 and S4 had both about 208 Kbps average bit rate in the delay analysis in order to point out treatment differences) and was only a small fraction of the handled traffic. In the delay analysis all voice and video sources with delay restrictions had reserved bandwidths above their average bandwidth since reservation has not a clear influence on packet delays because the out-of-profile packets are likely dropped and complicate the delay analysis when they go through. The background traffic was the largest part of the link traffic and was generated by 100 Web sources with the parameters being the same as for the Web sources described in Section 3.2. Three buffers–B1, B2, B3—were used in each node, with WRR weights of 5, 2 and 1, and with sizes of 60, 80 and 1200 Kb. For a 10 Mbps link capacity these buffer settings ensured maximum per hop delays of 9.375, 31.25 and 937.5 ms, respectively. *max_nh* values of 5, 9, 4 and 5 were used for S1, S2, S3 and S4 at link ingress, respectively; i.e. the packets originating from S1 and S4 reach their destination at the link's end, S2 packets have to travel a further 4 hops and S3 packets are extracted after the 4th node of the link.

Under this general setting five different scenarios aimed to cover a reasonably wide context range were simulated. In the first case the maximum remaining delays at link ingress were set to $max\_rd = 62.5$, 45, 125 and 75 ms for S1, S2, S3 and S4, respectively, in accordance with Reference [13]. A 10 Mbps forwarding capacity was considered for each of the five links. The mean and maximum delay values resulting from the simulation are given in the first row of Table I. The acceptance delay—calculated as (max_rd/max_nh) multiplied by the number of hops in the simulated link, e.g. 4 for S3—was added to the table in order to point out the operation of the algorithm. From this first case it is apparent that, as long as the traffic from sources with delay requirements is just a small fraction the forwarding capacity, the delay experienced by all sources is very small and there is little differentiation in treatment. It should be pointed out however that the slightly larger S3 and S4 delays resulted from the tendency to use the more occupied of the first two buffers for S3 and S4 packets.

Trying to induce some treatment differentiation, the second case considered forwarding capacities of 10, 2, 1, 1 and 10 Mbps, from ingress to egress, while keeping the rest of the simulation parameters unchanged. The resulted congestion induced a significant delay increase but the resulted delay values are still within the accepted margins with the exception of the maximum delay for S2, which has the tightest delay requirement. However, it can be observed that, despite being higher than the acceptable delay, the maximum delay is still kept in check by the algorithm since, as long as the minimum local delay is less than *max_rd* there is still a possibility to recover the supplementary local delay (in the worst case 28.03–25.0 ms) over the remaining hops.

In the third scenario the outbound link capacities were the same as in the second one but, additionally, the *max_rd* values were reduced below the mean or maximum delay achieved in the second scenario. What should be observed is that, when the acceptable delay per node cannot be met (min.delay $>$ *max_rd*/*max_nh*), the algorithm tries to keep the delay low only as long as there is a chance (min.delay $<$ *max_rd*) that the supplementary local delay can be compensated down the path. This is the situation for S2 packets, which are supposed to travel a further 4 hops. As soon as the maximum remaining delay is surpassed by the local minimum delay—i.e. there is no chance to meet the delay requirements—the algorithm, in our implementation, does no longer attempt to minimize the delay and treats those packets as packets without delay requirements. This is the situation for some of the packets of the S1 flow, which are treated as best effort packets over the last hop, thus increasing the maximum delay for S1 while the mean delay is still kept in range. It should be noted that the packets for which the delay requirements cannot be met may, alternatively, be dropped altogether depending on the application.

The facts that the average delay is reduced in all sources when tightening the requirements from Case 2 to Case 3 and that there is no average delay compensation (the reduced delay in some sources being compensated by increased delay in other sources) indicate that there are still unused resources that SWIFT has reserved 'just in case', despite the congestion. Therefore we have pushed further the congestion by simulating Cases 4 and 5 where the forwarding capacities were taken as 7.5, 1.5, 0.75, 0.75, 7.5 and 5, 1, 0.5, 0.5, 5 Mbps, respectively, while the rest of the simulation parameters were those of Case 2 except for the *max_rd* of S3, which kept its Case 3 value (see Table I).

The fact that the algorithm tries to satisfy delay requirements despite the capacity/congestion variations can be observed when comparing the delays of Case 4 and 2. It can be seen that the S3 delays are decreasing from Case 2 to Case 4 despite the increased congestion, because of the tighter Case 4 delay requirements.

In Case 5 the forwarding capacities and the delay requirements were brought to such values that the achievable delays are quite close to the requirements. Consequently, while the delays for S1 and S3 are just kept in range (with maximum delays only slightly below the acceptable values), the maximum delays for S2 and S4 soar due to the fact that some of their packets are treated as best effort when their delay requirements cannot be met. The mean delay for S4 remains below the acceptable delay and the mean delay for S2, despite surpassing the acceptable delay, remains below *max_rd*, thus keeping open the possibility to satisfy the delay requirement over the remaining hops.

By analysing the delays assured for similar sources (S1 and S2, S3 and S4), one can see that, besides obeying the requirements as long as it is possible, SWIFT ensures treatment differentiation over a wide range of conditions for relatively slight requirement variation.

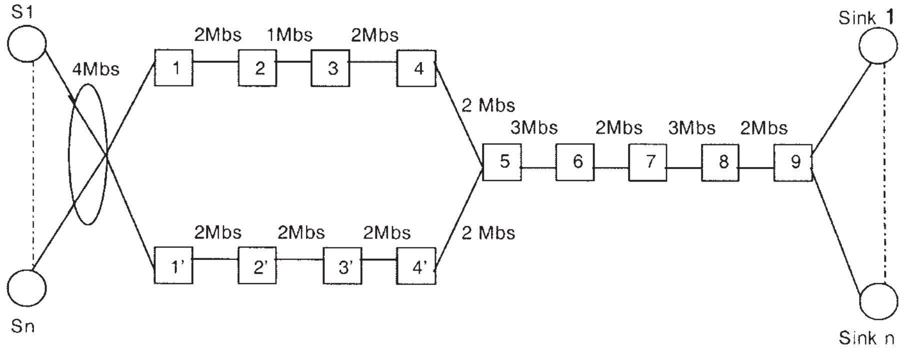

Figure 9. Link layout used for jitter simulation.

In conclusion for the delay analysis it should be underlined that SWIFT's strategy for delay management and differentiation is based on trying to provide each packet, if possible, a delay just below the local requirement (below *max_rd*/*max_nh*) and *not the minimum delay*. Consequently, if the capability for providing smaller delays exists, it is kept for other (subsequent) packets that might require it. Furthermore, when the local delay requirements cannot be met SWIFT still ensures the minimum local delay when there is a chance that the supplementary local delay can be compensated on the remaining path—thus enlarging the local delay optimization procedure to a global scope.

3.5. Jitter control

The SWIFT algorithm intrinsically aims at minimizing the jitter as it does not try to provide minimum delay but rather a delay as close as possible to the local delay requirement (*max_rd* /*max_nh*). If two packets with the same initial delay requirement experience significantly different delays over a part of their path they will have, as a consequence, different *max_rd*, which, in turn, will subsequently induce a smaller local delay requirement for the packet that previously experienced a larger delay. Thus the packets that were transferred relatively slower up to a point on their path are speeded up with respect to those packets that experienced a faster transfer.

In an attempt to show the intrinsic jitter control we simulated the situation in which the packets from various sources take two paths with different delay characteristics—one not congested, which induces a small delay and one congested, which induces a significantly greater delay—and then are transferred over a common link-path. The layout of the network structure simulated for jitter analysis is given in Figure 9. The 1-2-3-4 path was dubbed the 'slow' path while the 1′-2′-3′-4′ path was dubbed the 'fast' path. It should be mentioned that, paradoxically, the jitter control mechanism starts operating only when the remaining path on which the jitter is to be minimized is strongly solicited by the traffic having a reservation. This is because, when the traffic with delay reservation uses only a small fraction of the resources, the buffers reserved for packets with delay requirements are empty almost all the time, which means that all delay-bound packets experience similar low delay irrespective of their *max_rd*. Consequently, in our simulation the link capacities over the common 5-6-7-8-9 path were taken as 3-2-3-2 Mbps.

The sources used in the jitter analysis were two voice sources (S1 and S2 with the same parameters as those mentioned before), five video sources (S3, S4, S5, S6 and S7 derived from

Table II. Simulated average and worst-case jitter evaluation.

	Delay (ms)	After node 5	After node 7	After node 9
S1	Mean jitter	34.84	27.22	22.17
	Max jitter	77.37	61.48	47.01
S2	Mean jitter	43.45	36.15	28.92
	Max jitter	80.26	71.35	58.49
S3	Mean jitter	74.21	62.68	47.35
	Max jitter	119.48	97.32	76.11
S5	Mean jitter	66.58	51.18	42.34
	Max jitter	92.90	78.65	63.25

bursty video-conference traces of about 208 Kbps each) and several www best effort Web-like traffic sources, which amounted to about 3 Mbps and were used as background traffic. The traffic was equally divided among the 'slow' and 'fast' paths leading to a severe congestion on the 'slow' path while the 'fast' path had a transfer capacity close to the average transferred traffic thus leading to small congestion only during sources' bursts.

As can be seen from the example values given in Table II both the average jitter (calculated as the difference in mean delay between the packets that took the 'slow' path and those that took the 'fast' path) and the worst case jitter (calculated as the difference between the maximum delay for the packets that took the 'slow' path and the minimum delay for the packets that took the 'fast' path) are decreasing along the common 5-6-7-8-9 path for the traffic that can be kept within its delay requirements. It should be, however, underlined that, although the jitter reduction is a feature that derives from the principle of SWIFT delay management, the amount of jitter reduction is strongly dependent on the network status and on the range of buffer sizes and associated processing capabilities (i.e. on the availability of buffers with occupancies, sizes and associated processing capabilities providing a wide range of delays).

4. CONCLUSIONS AND FURTHER WORK

The novel 'Simple Weighted Integration of diFferentiated Traffic' (SWIFT) DiffServ approach was proposed and its implementation was tested by simulation over a relatively wide range of local network contexts. The resulted throughput, packet drop occurrence, mean delay, maximum delay and jitter were presented and SWIFT characteristics were analysed. Some comparisons with other approaches employing traffic control have been derived from simulations.

The simulation results show, among others, that SWIFT ensures: (a) isolation of the in-profile traffic from congestion effects, (b) treatment differentiation for relatively slight requirement variation over a wide range of conditions, (c) increased resource utilization, including the accommodation of out-of-profile and best effort traffic, (d) non-blocking conditions as long as resources are available, (e) fairness in treatment under congestion, (f) possibility to recover the effects of some local packet treatments that do not satisfy the service requirements and (g) incentivity for nice behaviour.

There are still further tests needed for assessing the capabilities of SWIFT in a more realistic context. A much greater number of sources, with a wider range of behaviours should be used to analyse the treatment differentiation and its granularity. The range of buffer dimensioning necessary for effective delay differentiation and jitter control should be determined in a broader range of network structure and traffic context. Also traffic with a wider range of delay requirements should be transferred through complex network configurations with many hops and under a broad range of congestion contexts to analyse the achievable performances in realistic conditions.

ACKNOWLEDGEMENTS

The work presented in this paper was funded by the Graduate School in Electronics, Telecommunications and Automation (GETA) and by the National Technology Agency of Finland (TEKES).

REFERENCES

1. Braden R, Clark D, Shenker S. *Integrated Services in Internet Architecture, an Overview*. *IETF RFC 1633*, October 1997.
2. Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. *An architecture for Differentiated Services*. *IETF RFC 2475*, December 1998.
3. Guerin R, Pla. V. Aggregation and conformance in differentiated service networks: A Case Study. *Computer Communication Review* 2001; **31**(1):21–32.
4. Xu Y, Guerin R. Individual QoS versus aggregate QoS: a loss performance study. *INFOCOM'2002*: 2002; (3): 1170–1179.
5. Stoica I, Shenker S, Zhang H. Core-stateless fair queuing: a scalable architecture to approximate fair bandwidth allocations in high speed networks. *Proceedings of ACM SIGCOMM*, September 1998; 118–130.
6. Stoica I, *et al.* Per Hop behaviours based on dynamic packet states. *Internet Draft*. draft-stoica-diffserv-dps00.txt; Feb. 1999.
7. Stoica I, Zhang H. Providing guaranteed services without per flow management. *Proceedings of SIGCOMM'99*, September 1999; 81–94.
8. Chuang ST, Goel A, McKeown N, Prabhakar B. Matching output queuing with a combined input–output queued switch. *INFOCOM'99*, March 1999; 1169–1178.
9. Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, August 1993; **1**(4):397–413.
10. Clark D, Fang W. Explicit allocation of best effort packet delivery service. *IEEE/ACM Transactions on Networking* 1998; **6**(4):362–373.
11. Crovella ME, Bestavros A. Explaining world wide web traffic self-similarity. Computer Science Dept., Boston University, *Technical Report* TR-95-015, August 1995.
12. Deng S. Empirical Model of WWW Documents arrival at access link. *Proceedings of ICC'96*. 1996; (3):1797–1802.
13. ITU. One-way transmission time. *Recommendation G.114*, February 1996.

AUTHORS' BIOGRAPHIES

Avadora Dumitrescu, received her MSc in Electrical Engineering, Telecommunications specialization, from the 'Politehnica' University of Bucharest in 1996. Currently she is a researcher at the Institute of Communications Engineering, Tampere University of Technology, Finland, working towards her PhD degree. Her research interests include performance modelling and analysis of communications networks with emphasis on QoS, congestion control and traffic management.

Jarmo Harju, received his MSc from Helsinki University of Technology in 1979 and PhD from the University of Helsinki in 1984. During 1985–1989 he was senior researcher at the Telecommunications Laboratory of the Technical Research Center of Finland, working with the development of protocol software. In 1989–1995 he was professor of data communications at Lappeenranta University of Technology. Since 1996 he has been professor of telecommunications at Tampere University of Technology, Institute of Communications Engineering, where he is leading the 'Networks and Protocols' group. His research interests include QoS mechanisms and congestion control in packet switched networks.