

# A Fair and Scalable Approach for Differentiated Services Networks

Avadora Dumitrescu, Jarmo Harju

Tampere University of Technology, Institute of Communication Engineering  
P.O. Box 553, FIN-33101, Tampere, Finland  
{avadora, harju}@cs.tut.fi

## Abstract:

In order to achieve a Quality of Service (QoS) capable of satisfying different user requirements, Differentiated Services (DiffServ) have been introduced as a scalable and viable solution in today's Internet. Mapping the packet treatment into a small number of Per Hop Behaviors (PHBs) clearly improves the scalability but it comes at the cost of losing some fairness between flows multiplexed into the same aggregated traffic. In this paper, a scalable approach for a fair and smaller 'granularity' in user differentiation is presented. The proposed architecture is evaluated and analyzed via simulations considering both simplified and more realistic traffic scenarios. Some comparison with other architectures employing traffic control is also provided.

## 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 was successful until now, when most of the traffic was TCP based, relying on end-to-end congestion mechanism that TCP provides. As a broad range of new applications competes now for the Internet resources, the traditional uniform treatment of all the packets is no longer able to cope with the emerging wide range of explicit requirements. Moreover some 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) and Differentiated Services (DiffServ). 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, and 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.

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. Upon entering a DiffServ domain, a packet is processed based on a small number of service levels or *Per Hop Behaviors* (PHBs). Mapping the flow's behavior into a small number of behavior classes clearly improves scalability but it comes at the cost of losing granularity and a great deal of fairness in end-to-end service delivery. Several studies, as those reported in [1],[2], analyze 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 and aggregate performance can exist. Consequently, the limited number of PHBs may induce similar treatment for flows with significant behavior differences.

In this paper, we propose a simple and effective DiffServ protocol, the 'Simple Weighted Integration of differentiated Traffic' (SWIFT) protocol, capable of delivering scalable services while preserving a fine granularity in service discrimination. As a starting point, we use the Dynamic Packet State (DSP) concept [3] in which the packet itself carries in its header some state that is initialized by the ingress router. The packet is processed (forwarded or dropped) based only on the information carried by its header and on the router's internal state.

## 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. The SWIFT algorithm has several components – congestion state determination, packet admission decision, packet buffering and packet information updating (re-labeling) – and uses DSP by encoding the information used for packet handling in the packet header. 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 SLA, while  $abr$  is measured 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 zero 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 comparing with the case when they would be encoded separately. The information about traffic requirements (delay and expected bandwidth) carried by packets themselves allows to achieve fine service granularity for traffic aggregate within a PHB, still preserving the scalability of DiffServ approach. Besides the information carried by the packet, the packet treatment takes into account also 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 behavior of output queuing routers [4] and since the output queuing architecture is easier to

analyze 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 (due to some possible 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  $Buff\_Occup/C_i$ . 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  $max\_rd$ , which means 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 'color' marker (with the 'classical' green, yellow and red colors) to describe its congestion state. The 'color' 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 pre-determined 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. Since the momentary capability to handle incoming flows is related to both the capacity of the outgoing link and the existing buffer space, in SWIFT, the forwarding capability is calculated by multiplying/ 'weighting' the forwarding capacity with the buffer(s) occupancy ratio. Furthermore, in evaluating the forwarding requirement of the existing traffic at the moment when a certain

packet is processed, the EMA-averaged aggregate traffic is multiplied/'weighted' by the  $abr/rsv$  ratio of the processed packet. This is done to make the packet admission decision assuming that the whole traffic is behaving as good or as bad as the flow to which the packet belongs to. Consequently the comparison on which the packet admission decision is taken under congestion is between  $C*(BuffSize-BuffOcc-BuffMarg)/BuffSize$  and  $(abr/rsv)*Agtf$ , where  $C$  is the outgoing link capacity,  $BuffSize$  is the buffer size,  $BuffOcc$  is the buffer occupancy,  $BuffMarg$  is a buffer margin and  $Agtf$  is the exponential moving average of the aggregated traffic.

Depending on the congestion status the abovementioned comparison is made at different levels. For the 'yellow' congestion level the forwarding capacity, buffer occupancy ratio and aggregated traffic are considered for the whole node while for the 'red' (congestion) 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 momentary forwarding capability,  $C*(BuffSize-BuffOcc-BuffMarg)/BuffSize$ , is bigger than  $(abr/rsv)*Agtf$  the packet is accepted. In order to accommodate some level of burstiness and/or random variations of the flow parameters at packet level, the packets are not dropped altogether when  $C*(BuffSize-BuffOcc-BuffMarg)/BuffSize$  is smaller than  $(abr/rsv)*Agtf$  but they are still forwarded with a probability:  $fp = \min \{ 1, (rsv/abr)*(BuffSize-BuffOcc)/BuffSize \}$ , i.e. 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. The use of the forwarding probability ( $fp$ ) enables a smooth interface 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 transfer. 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  $C*(BuffSize-BuffOcc-BuffMarg)/BuffSize$  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  $C*(BuffSize-BuffOcc-BuffMarg)/BuffSize$  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 before the packet is effectively forwarded. 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 zero, 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 also a null  $max\_rd$  since they also do not have a delay requirement. 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  $abr/rsv$  factor in the packet admission decision is replaced, when best effort packets are treated, with the maximum  $abr/rsv$  value for the traffic having a reservation, multiplied by a supra-unitary correction factor. In our implementation the correction factor has a given supra-unitary value (for example 1.5) as long as the maximum  $abr/rsv$  has a sub-unitary value (i.e. the traffic having a reservation is well-behaved). When the maximum  $abr/rsv$  increases above one (i.e. there is at least one badly behaving flow) the correction factor asymptotically decreases to one as  $abr/rsv$  increases. This dependence of the correction factor on the supra-unitary  $abr/rsv$  was chosen in order to diminish the advantage given to reserved traffic with respect to best effort traffic as the reserved traffic is behaving worse. The correction function could also be defined to decrease to 1 or even below 1 for high supra-unitary  $abr/rsv$  in order to discourage extremely malicious behavior and to eliminate the possibility that a single very badly behaving flow with some reservation deteriorates the treatment of all the best effort traffic. The forwarding probability function used for best effort traffic is also altered using a subunitary factor  $\alpha$ :  
$$fp = \min\{ \alpha, (\alpha * \max(rsv/abr) * (BuffSize - BuffOcc) / BuffSize) \}$$

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 maximum  $abr/rsv$  value is continuously updated over a certain time window and at the beginning of a new time window it is initialized with half the value obtained over the previous 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

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

In a first simplified scenario three sources, S1, S2 and S3 compete for the forwarding capacity of a node both under congestion and without congestion. S1 is a constant bit rate (CBR) source of 4Mbps with a reserved bandwidth that varies. S2 is an ON-OFF source with a 4Mbps bit rate in the ON state and a bandwidth reservation of 4Mbps. Finally S3 is a CBR source of 4Mbps with no bandwidth reservation (i.e. best effort traffic). The node forwarding capacity is 10Mbps 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 considered T1=0.1 and T2=0.6.

For buffer size dimensioning we have considered a maximum acceptable per node delay of 75 ms, which corresponds to 96 KB 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 Fig. 1, is very different from source to source and strongly dependent on congestion. S2 which is ‘well-behaved’ 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 2Mbps for a constant bit rate of 4Mbps, 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.

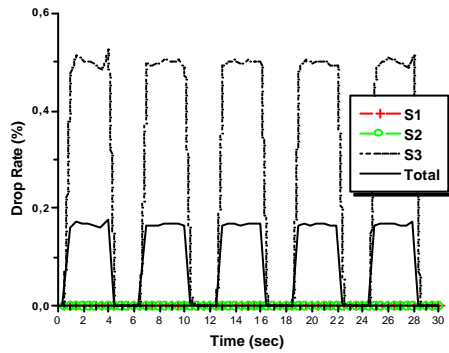


Fig. 1. Drop rates variation in time

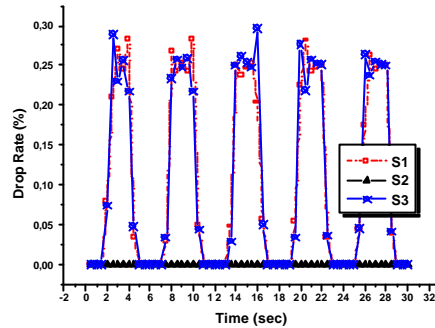


Fig. 2. Drop rates for S1 and S3 when they are competing as best effort sources

Fig. 2 shows that, when the reservation of S1 drops to zero, thus putting both S1 and S3 in 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.

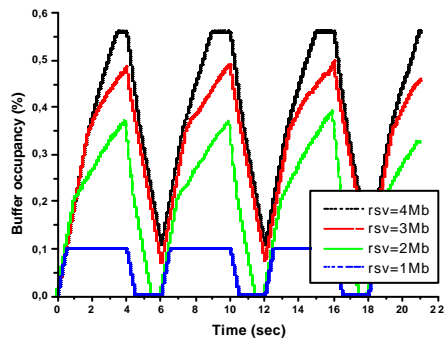


Fig. 3. Overall buffer occupancy variation with S1 reservation as a parameter

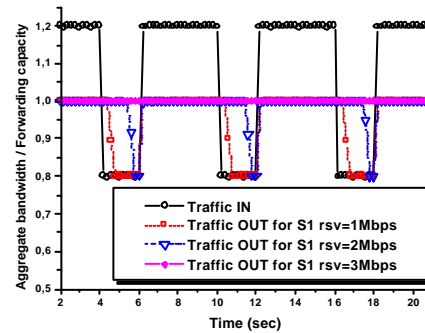


Fig. 4. Output aggregate bandwidth variation with S1 reservation as a parameter

Fig. 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 node capacity.

For large reservation values for S1 ( $rsv=3\text{Mbps}$  and  $4\text{Mbps}$ ), 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 Fig. 4.

### 3.1 Comparison with Other Approaches

In this section we evaluate the SWIFT algorithm by comparing it with some other existing approaches. The previously described simulation scenario is considered (i.e three sources – S1, S2, S3 – sending traffic through a 10 Mbps node) 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 discipline without any congestion mechanism control, 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 [5] 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 weight factor ( $wq$ ) that influences the calculation of the average queue size.
- RIO (Random Early Detection with In and Out) – Based on RED mechanism, RIO [6] introduces an additional complexity in buffer management by discriminating against 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  $Pmax=0.02$ ,  $wq=0.002$ ,  $minth=0.1$  and  $maxth=0.6$  [s3], 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 behavior 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.

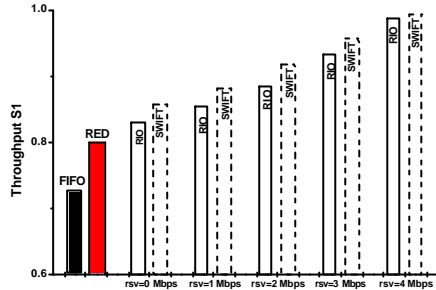


Fig. 5. Throughput for S1 when using FIFO, RED, RIO and SWIFT– for RIO and SWIFT the throughput is shown for several S1 reservations.

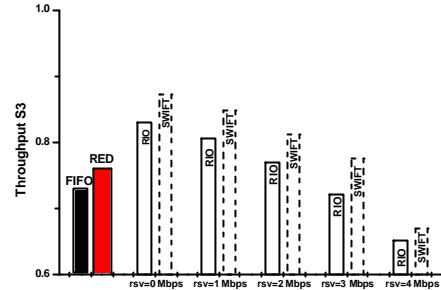


Fig. 6. Throughput for S3 when using FIFO, RED, RIO and SWIFT– for RIO and SWIFT the throughput is shown for several S1 reservations.

Fig. 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.

When looking at the variations of S3 throughput in Fig. 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 can 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.

### 3.2 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 125Kbps, modeled according to [7] and [8]. The sources are ON-OFF with the ON-OFF periods drawn from heavy tailed Weibull and Pareto distribution,

respectively. The packet inter-arrival times corresponding to the ON period follow 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 13Kbps and 32Kbps bit rates involving GSM and ADPCM coding and compression, respectively. In our simulations S1 had a 15Kbps 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 220Kbps reservation, was a bursty one with an average bandwidth of 208Kbps. The trace for S4, for which no reservation was considered, was also quite bursty, with an average bandwidth of 135Kbps. The node forwarding capacity was taken to be just 1Mbps 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 wwww traffic on equal footing.

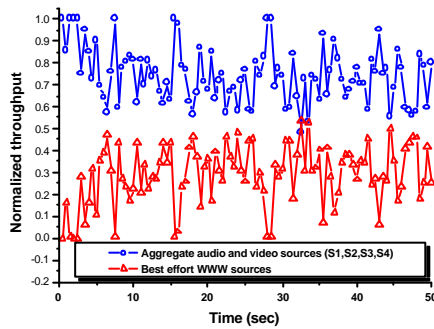


Fig. 7. Normalized throughput of the aggregate audio and video traffic compared with that of the aggregate wwww best-effort traffic.

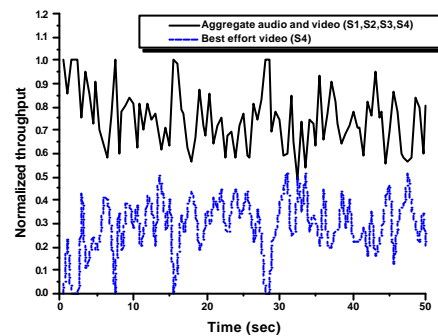


Fig. 8. Normalized throughput for the aggregate audio and video traffic compared with that of the best-effort video source (S4)

Fig. 7 shows the aggregate throughput of the audio and video sources (S1, S2, S3 and S4) and the aggregated wwww traffic throughput. It can be observed that the two variations are somehow in 'antiphase' due to the 'opportunistic' transfer of the wwww traffic under congestion – i.e. whenever the traffic with a reservation has a burst, thus increasing the aggregate audio and video traffic, the wwww traffic is cut down and whenever the reserved traffic has a drop in bandwidth the wwww 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.

Fig. 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 wwww traffic throughput variation of Fig. 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 Fig. 7 and 8 and can also be deduced from the throughput budget of Fig. 7:  $13\text{Kbps} + 208\text{Kbps} + 0.3 * (32\text{Kbps} + 135\text{Kbps}) \approx 0.7 * (13\text{Kbps} + 208\text{Kbps} + 32\text{Kbps} + 135\text{Kbps})$ . Fig. 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 processing capacity.

### 3.3 Delay Analysis

In order to evaluate SWIFT's capacity to assure delay requirements we have simulated a one-way transfer over a congested five-node link. The traffic with delay restrictions came from the previously described voice sources (S1 and S2 – with 13Kbps and 32Kbps average bit rate, respectively) and video sources (S3 and S4 – with 208Kbps and 135Kbps average bit rate, respectively) and was only a small fraction of the traffic handled by the link. In the delay analysis all sources with delay restrictions had reserved bandwidths above their average bandwidth since reservation has not a significant influence on packet delays. 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, ensuring maximum per hop delay of 9.375, 30.518 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 4<sup>th</sup> node of the link.

Under this general setting three different scenarios aimed to cover a rather wide parameter 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, and a 10Mbps forwarding capacity was considered for each of the five nodes. Resulting from the *max\_nh* values the acceptable delay per hop and the acceptable delay over the five node link (*acc* in Table 1, calculated as the number of hops in the simulated link – 4 for S3 – multiplied by *max\_rd* / *max\_nh*) were compared with the mean and maximum delays resulted from simulation. Due to the fact that in this scenario the traffic from sources with delay requirements was just a small fraction of each capacity, the delay experienced by all sources was very small. Taking into account the results of the simulation for this first case (given in the first line of Table 1) a second scenario was devised, where the forwarding capacities were taken as 10, 2, 1, 1 and 10 Mbps, from ingress to egress, while keeping the rest of the simulation parameters unchanged.

Link delay	S1			S2			S3			S4		
	acc.5	mean	max	acc.9	mean	max	acc.4	mean	max	acc.5	mean	max
Case 1	62.50	2.72	4.08	25.00	2.94	4.31	125.0	4.83	8.55	75.0	3.03	5.56
Case 2	62.50	21.66	42.44	25.00	20.03	28.03	125.0	28.84	80.74	75.0	24.84	70.67
Case 3	18.00	18.23	59.35	15.00	14.84	28.94	80.0	20.97	70.18	65.0	21.32	68.81

Table 1. Accepted and simulated delay (in ms), over the whole five-node link, for S1, S2, S3 and S4, in different simulation scenarios.

Under the second scenario the 1 Mbps links limited the packet transfer speed increasing the delays as can be seen in the second row of Table 1. It can be seen that the resulted mean and maximum delays increase for bigger *max\_rd*. For S2, where the delay requirements are the smallest, the maximum delay over the whole five-node link surpasses the acceptable delay while the mean delay remains in range, indicating that for some packets the delay requirement cannot be met, but for which SWIFT keeps offering the minimum delay in the hope that the supplementary local delay (in the worst case 28.03-25.0) might be compensated over the remaining 4 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 can be seen from the third scenario simulation results is that, when the maximum delay per node/link cannot be met ( $\text{min.delay} > \text{max\_rd} / \text{max\_nh}$ ), the algorithm tries to keep the delay low only as long as there is a chance ( $\text{min.delay} < \text{max\_rd}$ ) that the supplementary delay can be compensated down the path. As soon as the maximum remaining delay is surpassed by the local minimum delay the algorithm does no longer attempt to minimize the delay and, in our implementation, treats those packets as packets without delay requirements. Thus the delay for some of the packets of the flow becomes very large although the mean delay is still kept in range (for S1 and S4). The packets for which the delay requirements could not be met may be, alternatively, dropped altogether depending on the application.

The results somehow illustrate SWIFT's strategy for delay management, which is to provide each packet, if possible, a delay just below the local requirement (below  $\text{max\_rd} / \text{max\_nh}$ ) and *not* the minimum delay. Thus the capability for providing smaller delays is kept for other (subsequent) packets that might require it.

#### **4. CONCLUSIONS AND FURTHER WORK**

The 'Simple Weighted Integration of differentiated Traffic' (SWIFT) DiffServ protocol was implemented and some relatively simple simulations, chosen so that the adequate behavior of SWIFT can be easily judged, were carried out to analyze and assess its capabilities. The performed simulations proved some important characteristics of SWIFT. SWIFT insulates the traffic having some reservation from the effects of congestion as long as the congestion can be solved by dropping packets with lower priority. In the meantime none of the least prioritized traffic flows is blocked unless the reserved traffic bursts fill-up the capacity. Also the traffic with the same priority is fairly treated, experiencing the same drop rate.

SWIFT assures delay differentiation by attempting to satisfy local delay requirements with certain margins. This delay treatment approach ensures, opposed to the situation when minimum delay is aimed, that faster-than-required forwarding capabilities are kept in reserve for other packets with more stringent delay requirements. In the meantime, SWIFT attempts to assure minimum delay, even if the local delay requirements cannot be met, as long as there is a possibility that the supplementary local delay can be compensated later on the packet path.

There is still a great amount of further tests needed for assessing the capabilities of SWIFT in a realistic network and traffic structure. A much greater number of sources, with a wider range of behaviors should be used to analyze the treatment differentiation and its

granularity. 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 analyze the achievable delay performances in realistic conditions.

## REFERENCES

- [1] Guerin R., and V. Pla., 2001, "Aggregation and Conformance in Differentiated Service Networks: A Case Study.", *Computer Communication Review*, Vol. 31, No. 1, January, pp. 21-32
- [2] Xu Y., and R. Guerin. 2001. "Individual QoS versus aggregate QoS: A loss performance study." Technical Report, University of Pennsylvania, July (accepted for presentation at INFOCOM'2002, New York, NY, June 2002).
- [3] Stoica I.et. al. 1999. "Per Hop Behaviors Based on Dynamic Packet States", Internet Draft, draft-stoica-diffserv-dps00.txt, Feb.
- [4] Chuang S.T., A. Goel, N. McKeown, B. Prabhakar. 1999. "Matching output queueing with a combined input-output queued switch", *Proceedings of INFOCOM'99*, pp. 1169-1178, New York, March .
- [5] Floyd S. and V. Jacobson. 1993. " Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, (August) pp. 397-413.
- [6] Clark D. and W. Fang. 1998. "Explicit Allocation of Best Effort Packet Delivery Service", *IEEE/ACM Transactions on Networking* 6, no. 4 , pp 362-373
- [7]Crovella M.E. and A. Bestavros.1995. "Explaining World Wide Web Traffic Self-Similarity", *Computer Science Dept., Boston University, Technical Report TR-95-015*, August 29.
- [8] Deng S., 1996. "Empirical Model of WWW Documents Arrival at Access Link' , *ICC'96*, vol. 3, page 1797-1802