Annales Univ. Sci. Budapest., Sect. Comp. 25 (2005) 85-102

SIMULATION OF DIFFERENTIATED SERVICES IN NETWORK SIMULATOR

M. Lengyel (Debrecen, Hungary)
J. Sztrik (Debrecen, Hungary)
C.S. Kim (Wonju, Korea)

Abstract. The history of the Internet has been of continuous growth in the number of hosts, the number and variety of applications, and the capacity of the network infrastructure. A scalable architecture for service differentiation must be able to accommodate this continuous growth. The Differentiated Services (Diffserv or DS) architecture provides a more flexible, scalable architecture than the existing models of service differentiation. The specification of Diffserv architecture appeared in 1998, but the current research is still expanding it. In this paper, we simulate Diffserv networks in Network Simulator (NS) making throughput, delay, queue length comparison in case when within a simple network topology different schedulers are used. The traffic generator is Constant Bit Rate (CBR) over User Datagram Protocol (UDP). We analyze how the modification of the Random Early Detection (RED) parameters influence the above mentioned performance measures.

1. Introduction

The Differentiated Services architecture [1] defines a model for implementing scalable service differentiation in the Internet. The architecture is based on a simple model where traffic entering a network is classified and possibly

The research was partially supported by the Hungarian National Foundation for Scientific Research under grant T034280/2000, the FKFP grant 0191/2001 and the KOSEF-Hungarian Academy of Sciences Bilateral Scientific Cooperation (grant KOSEF F01-2004-000-100510-0), 2004.

conditioned at the boundaries of the network, and assigned to different DS codepoints. Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DS codepoint. A per-hop behavior (PHB) is a description of the externally observably forwarding behavior of a DS node applied to packets with a particular DS codepoint. PHBs are implemented in nodes by means of some buffer management and packet scheduling mechanisms. Two different PHBs were developed: the Assured Forwarding (AF) PHB [2] and the Expedited Forwarding (EF) PHB [3]. The AF PHB group provides delivery of IP packets in four independently forwarded AF classes. Within each AF class, an IP packet can be assigned to one of three different levels of drop precedence. EF PHB is intended to prove low delay, low jitter and low loss services by ensuring that the EF packets are served at a certain configured rate.

Diffserv is composed of a number of functional elements implemented in network nodes, including a small set of per-hop forwarding behaviors, packet classification functions, and traffic conditioning functions including metering, marking, shaping, and policing. This architecture achieves scalability by implementing complex classification and conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregates of traffic which have been appropriately marked using the DS field in the IPv4 or IPv6 headers. Per-hop behaviors are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams.

This architecture only provides service differentiation in one direction of traffic flow and is therefore asymmetric.

While many studies have addressed issues on the Diffserv architecture (e.g. dropper, marker, classifier and shaper), there have been few attempts to analytically understand a flow's behavior in a diffserv network.

In this paper we consider a simple dumbbell Diffserv network topology in which performance comparison (in terms of throughput, delay and queue length) is made between the traffic scheduling algorithms: Priority (PRI), Weighted Round Robin (WRR) and Weighted Interleaved Round Robin(WIRR) schedulers. Random Early Detection (RED) [4], [5] is used as active queue management algorithm. The novelty of the paper is these comparisons since to the best knowledge of the authors in the earlier papers only individual schedulers were analyzed.

In the core of the network there is a bottleneck link and the consideration is performed on that node. All of our traffic generators are Constant Bit Rate (CBR), the transport protocol is User Datagram Protocol (UDP). We used Network Simulator [8] (NS, version 2) for our simulation experiments.

In Section 2 we first discuss the architectural model of the Diffserv. Section 3 describes the RED active queue management algorithm. Results of the simulation are presented in Section 4. Conclusions are drawn in Section 5.

2. Differentiated Services scheme

For easier understanding of the results presented in Section 4 a brief description of the Diffserv architecture is provided in this part of the paper.

A *DS domain* is a contiguous set of DS nodes which operate with a common service provisioning policy and set of PHB groups implemented on each node. A DS domain has a well-defined boundary consisting of DS boundary nodes which classify and possibly condition ingress traffic to ensure that packets which transit the domain are appropriately marked to select a PHB from one of the PHB groups supported within the domain. Nodes within the DS domain select the forwarding behavior for packets based on their DS codepoint, mapping that value to one of the supported PHBs using either the recommended codepoint $\rightarrow$ PHB mapping or a locally customized mapping. A DS domain normally consists of one or more networks under the same administration; for example an organization's intranet.

A DS domain consists of *DS boundary nodes* and *DS interior nodes*. DS boundary nodes interconnect the DS domain to other DS or non-DS-capable domains, whilst DS interior nodes only connect to other DS interior or boundary nodes within the same DS domain. Both DS boundary nodes and interior nodes must be able to apply the appropriate PHB to packets based on the DS codepoint. In addition, DS boundary nodes may be required to perform traffic conditioning functions as defined by a traffic conditioning agreement (TCA) between their DS domain and the peering domain which they connect to. Interior nodes may be able to perform limited traffic conditioning functions such as DS codepoint re-marking.

DS boundary nodes act both as a *DS ingress node* and as a *DS egress node* for different directions of traffic. Traffic enters a DS domain at a DS ingress node and leaves a DS domain at a DS egress node. A DS ingress node is responsible for ensuring that the traffic entering the DS domain conforms to any TCA between it and the other domain to which the ingress node is connected. A DS egress node may perform traffic conditioning functions on traffic forwarded to a directly connected peering domain, depending on the details of the TCA between the two domains.

A *DS region* is a set of one or more contiguous DS domains. DS regions are capable of supporting differentiated services along paths which span the domains within the region. The DS domains in a DS region may support different PHB groups internally and different codepoint → PHB mappings. However, to permit services which span across the domains, the peering DS domains must each establish a peering service level agreement (SLA) which defines (either explicitly or implicitly) a TCA which specifies how transit traffic from one DS domain to another is conditioned at the boundary between the two DS domains. It is possible that several DS domains within a DS region may adopt a common service provisioning policy and may support a common set of PHB groups and codepoint mappings, thus eliminating the need for traffic conditioning between those DS domains.

The packet *classification* policy identifies the subset of traffic which may receive a differentiated service by being conditioned and/or mapped to one or more DS codepoint within the DS domain. Packet classifiers select packets in a traffic stream based on the content of some portion of the packet header. There are two types of classifiers. The BA (Behavior Aggregate) classifier classifies packets based on the DS codepoint only. The MF (Multi-Field) classifier selects packets based on the value of a combination of one or more header fields, such as source address, destination address, DS field, protocol ID, source port and destination port numbers, and other information such as incoming interface.

A traffic *profile* specifies the temporal properties of a traffic stream selected by a classifier. It provides rules for determining whether a particular packet is in-profile or out-of-profile. The concept of in- and out-of-profile can be extended to more than two levels, e.g. multiple levels of conformance with a profile may be defined and enforced. Different conditioning actions may be applied to the in-profile packets and out-of-profile packets, or different accounting actions may be triggered.

Traffic *conditioning* performs metering, shaping, policing and/or re-marking to ensure that the traffic entering the DS domain conforms to the rules specified in the TCA, in accordance with the domain's service provisioning policy. The extent of traffic conditioning required is dependent on the specifics of the service offering, and may range from simple codepoint re-marking to complex policing and shaping operations. A traffic stream is selected by a classifier, which steers the packets to a logical instance of a traffic conditioner.

A *meter* is used (where appropriate) to measure the traffic stream against a traffic profile. The state of the meter with respect to a particular packet (e.g. whether it is in- or out-of-profile) may be used to affect a marking, dropping or shaping action. Note that a traffic conditioner may not necessarily contain all four elements.

Packet *markers* set the DS field of a packet to a particular codepoint. When the marker changes the codepoint in a packet it is said to have "re-marked" the packet.

Shapers delay some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. A shaper usually has a finite-size buffer, and packets may be discarded if there is not sufficient buffer space to hold the delayed packets.

Droppers discard some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. This process is known as "policing" the stream. Note that a dropper can be implemented as a special case of a shaper by setting the shaper buffer size to zero (or a few) packets.

Traffic conditioners and classifiers are usually located *within DS ingress and egress boundary nodes*, but may also be located in *nodes within the interior of a DS domain*, or *within a non-DS-capable domain.*

The *PHB* is the means by which a node allocates resources to behavior aggregates, and it is on top of this basic hop-by-hop resource allocation mechanism that useful differentiated services may be constructed. PHBs may be specified in terms of their resource (e.g. buffer, bandwidth) priority relative to other PHBs, or in terms of their relative observable traffic characteristics (e.g. delay, loss). These PHBs may be used as building blocks to allocate resources and should be specified as a group (PHB group) for consistency. PHB groups will usually share a common constraint applying to each PHB within the group, such as a packet scheduling or buffer management policy. The relationship between PHBs in a group may be in terms of absolute or relative priority (e.g. discard priority by means of deterministic or stochastic thresholds), but this is not required (e.g. N equal link shares). A single PHB defined in isolation is a special case of a PHB group.

PHBs are implemented in nodes by means of some buffer management and packet scheduling mechanisms. PHBs are defined in terms of behavior characteristics relevant to service provisioning policies, and not in terms of particular implementation mechanisms. In general, a variety of implementation mechanisms may be suitable for implementing a particular PHB group. Furthermore, it is likely that more than one PHB group may be implemented on a node and utilized within a domain. PHB groups should be defined such that the proper resource allocation between groups can be inferred, and integrated mechanisms can be implemented which can simultaneously support two or more groups.

A PHB is selected at a node by a mapping of the DS codepoint in a received packet. Standardized PHBs have a recommended codepoint. However, the total space of codepoints is larger than the space available for recommended

codepoints for standardized PHBs, and the DS field specification leaves provisions for locally configurable mappings. A codepoint → PHB mapping table may contain both 1→1 and $N \rightarrow 1$ mappings. All codepoints must be mapped to some PHB; in the absence of some local policy codepoints which are not mapped to a standardized PHB in accordance with that PHB's specification should be mapped to the Default PHB.

Assured Forwarding (AF) PHB Group: It provides forwarding of IP packets in N independent AF classes. Within each AF class an IP packet is assigned one of M different levels of drop precedence. An IP packet that belongs to an AF class i and has drop precedence j is marked with AF codepoint AFij, where $1 <= i <= N$ and $1 <= j <= M$. Currently, four classes ($N = 4$) with three levels of drop precedence in each class ($M = 3$) are defined for general use. More AF classes or levels of drop precedence may be defined for local use. In case of congestion the drop precedence of a packet determines the relative importance of the packet within the AF class. A congested DS node tries to protect packets with a lower precedence value from being lost by preferably discarding packets with a higher drop precedence value. In a DS node the level of forwarding assurance of an IP packet thus depends on (1) how much forwarding resources has been allocated to the AF class that the packet belongs to, (2) what is the current load of the AF class, and, in case of congestion within the class, (3) what is the drop precedence of the packet.

A DS node should implement all four general use AF classes. Packets in one AF class must be forwarded independently from packets in another AF class, i.e. a DS node must not aggregate two or more AF classes together. A DS node must allocate a configurable, minimum amount of forwarding resources (buffer space and bandwidth) to each implemented AF class. Each class should be serviced in a manner to achieve the configured service rate (bandwidth) over both small and large time scales. An otherwise DS-compliant node is not required to implement this PHB group in order to be considered DS-compliant.

An AF class may also be configurable to receive more forwarding resources than the minimum when excess resources are available either from other AF classes or from other PHB groups.

Within an AF class a DS node must not forward an IP packet with smaller probability if it contains a drop precedence value p than if it contains a drop precedence value q when $p < q$. Note that this requirement can be fulfilled without needing to dequeue and discard already-queued packets. Within each AF class a DS node must accept all three drop precedence codepoints and they must yield at least two different levels of loss probability. If a DS node only implements two different levels of loss probability for an AF class x, the codepoint AFx1 must yield the lower loss probability and the codepoints AFx2 and AFx3 must yield the higher loss probability.

A DS node must not reorder AF packets of the same microflow when they belong to the same AF class regardless of their drop precedence. There are no quantifiable timing requirements (delay or delay variation) associated with the forwarding of AF packets. The AF PHB group may be used to implement both end-to-end and domain edge-to-domain edge services.

An AF implementation must attempt to minimize long-term congestion within each class, while allowing short-term congestion resulting from bursts. This requires an active queue management algorithm. An example of such an algorithm is *Random Early Drop* (RED). An AF implementation must detect and respond to long-term congestion within each class by dropping packets, while handling short-term congestion (packet bursts) by queueing packets.

To allow the AF PHB to be used in many different operating environments, the dropping algorithm control parameters must be independently configurable for each packet drop precedence and for each AF class. Other PHB groups may co-exist with the AF PHB group within the same DS domain.

Expedited Forwarding (EF) PHB: It is intended to provide a building block for low delay, low jitter and low loss services by ensuring that the EF aggregate is served at a certain configured rate. The dominant causes of delay in packet networks are fixed propagation delays (e.g. those arising from speed-of-light delays) on wide area links and queueing delays in switches and routers. Since propagation delays are a fixed property of the topology, delay and jitter are minimized when queueing delays are minimized. In this context jitter is defined as the variation between maximum and minimum delay. The intent of the EF PHB is to provide a PHB in which suitably marked packets usually encounter short or empty queues. Furthermore, if queues remain short relative to the buffer space available, packet loss is also kept to a minimum. To ensure that queues encountered by EF packets are usually short, it is necessary to ensure that the service rate of EF packets on a given output interface exceeds their arrival rate at that interface over long and short time intervals, independent of the load of other (non-EF) traffic. This is a PHB in which EF packets are guaranteed to receive service at or above a configured rate and provides a means to quantify the maximum delay and jitter that a packet may experience under bounded operating conditions.

The EF PHB is not a mandatory part of the DS architecture - a node is not required to implement the EF PHB in order to be considered DS-compliant.

Intuitively, the definition of EF is simple: the rate at which traffic is served at a given output interface should be at least the configured rate R, over a suitably defined interval, independent of the offered load of non-EP traffic to that interface. This intuition is formalized in [3].

Packets belonging to a single microflow within the EF aggregate passing through a device should not experience re-ordering in normal operation of the device.

Packets marked for EF PHB may be remarked at a DS domain boundary only to other codepoints that satisfy the EF PHB. Packets marked for EF PHBs should not be demoted or promoted to another PHB by a DS domain. Other PHBs and PHB groups may be deployed in the same DS node or domain with the EF PHB.

3. Active queue management

Random Early Detection (RED) is an active queue management algorithm for routers that will provide Internet performance advatages [4].

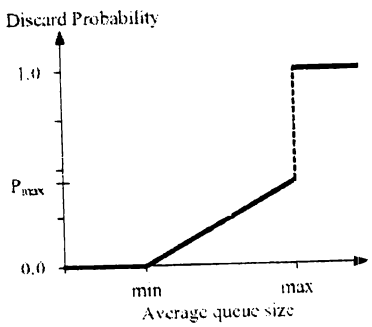


Figure 1. RED packet discard ratio

In contrast to traditional queue management algorithms, which drop packets only when the buffer is full, the RED algorithm drops arriving packets probabistically. The probability of drop increases as the estimated average queue size grows. Note that RED responds to a time-averaged queue length, not an istantaneous one. The RED algorithm itself consists of two main parts: *estimation of the average queue size* and *the decision of whether or not to drop an incoming packet.*

- Estimation of Average Queue Size: RED estimates the average queue size, either in the forwarding path using a simple exponentially weighted moving average, or in the background (i.e. not in the forwarding path) using a

similar mechanism. Note that the queue size can be measured either in units of packets or of bytes.

- Packet Drop Decision: In the second portion of the algorithm, RED decides whether or not to drop an incoming packet. It is RED's particular algorithm for dropping that results in performance improvement for responsive flows. Two RED parameters, *minth* (minimum threshold) and *maxth* (maximum threshold), figure prominently in this decision process. Minth specifies the average queue size below which no packets will be dropped, while maxth specifies the average queue size above which all packets will be dropped. As the average queue size varies from minth to maxth, packets will be dropped with a probability that varies linearly from 0 to maxth.

  Note that the decision whether or not to drop an incoming packet can be made in "packet mode", ignoring packet sizes, or in "byte mode", taking into account the size of the incoming packet.

All avilable empirical evidence shows that the deployment of active queue management mechanisms in the Internet would have substantial performance benefits.

4. Simulation results

Our simulation was performed using Network Simulator [8] (NS, version 2.1b9a), which was developed at the University of California. NS is an event-driven network simulator, which is implemented in C++ and uses OTcl (Object Tool Command Language) as the command and configuration interface. A simulation is defined by an OTcl script. The scripts use the Simulator Class as the principal interface to the simulation engine. Using the methods defined in this class we can define the network topology, we can configure the sources and the sinks, etc. NS is a free simulator software, it is continuously developed. NS does not have graphical user interface, it contains many different modules, so we can simulate a lot of network related topics.

We considered the simple dumbbell topology shown in Fig. 2. All links have the same fixed delay of 5 ms. The consideration is performed on the Core node where there is the bottleneck link. The link buffer has a capacity of 50 packets. The packet size is 1000 bytes in our simulations.

We ran each simulation for 60 seconds. This is a steady state because we compared it with the results of the 30 seconds run and there is no changing in results. The traffic generators are CBRs over UDP. We use only UDP protocol, because the TCP includes congestion avoidance methods which make

the analysis more complex. We make throughput, delay and queue length comparison between the scheduling algorithms PRI, WRR and WIRR. The nodes 0-3 generate AF1-AF4 traffic, while node 4 generates EF. The i-th node sends packets to the $i+8$-th node, $i = 0, 1, 2, 3, 4$. An AF class is implemented in the nodes as a RED physical queue with three virtual queues, while EF as a droptail (FIFO) queue. The structure of the output interface of the core node is shown in Fig. 3/a.

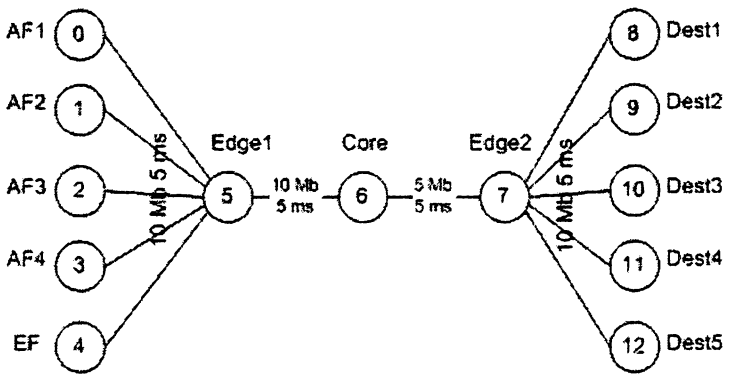


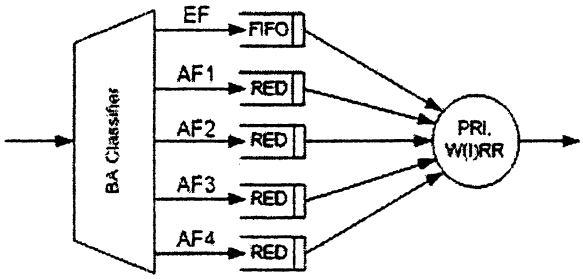
Figure 2. Simulated network topology

This is not the best solution for Diffserv networks implementation, a more flexible structure can be found in Fig. 3/b, but currently it is not supported by the NS, only if we extend the Diffserv implementation of the NS in this way.

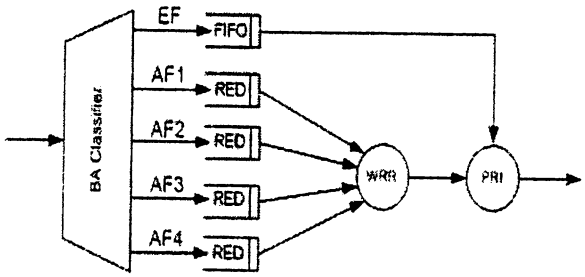
We set up the simulation scripts such that the Sent packet ratio (Fig. 4) and Received packet ratio (Fig. 5) is equal regardless of which scheduler is used.

The rates (percentages) of EF, AF1, AF2, AF3, AF4 classes in Fig. 4 depend on the sending rates of the CBRs in the scripts. The (percentages) distribution of the three subclasses (drop precedence) within each AF class is proportional with the size of the intervals defined by the parameters 0, *cir*, *pir*, *rate*. The parameters cir and pir are used as Committed information rate and Peak information rate value for the Time Sliding Window with 3 Color Marking (TSW3CM) policer, the rate parameter is the sending rate of the nodes (CBRs).

The rates of classes shown in Fig. 5 are calculated from the trace files. We get the same results if we take the sending rates, the 53% packet loss rate (be-



a) Simulated output interface



b) Ideal output interface

Figure 3. Output interface

cause the total sending rate configured in the script is 9.5 Mb/s and the bottleneck link is 5 Mb/s) and packet loss statistics by classes (Fig. 6) provided by software.

Fig. 6 shows the packet loss statistics by class. We can see that within each AF class the AFi3 has greater (or equal) packet loss ratio than AFi2, and AFi2 has greater (or equal) packet loss ratio than AFi1, $i = 1, 2, 3, 4$. This is what we expected, because we set up the RED parameters in the scripts in such way that the lower drop precedence class we give lenient RED parameter values.

Three figures show how the queue length varies in case of the three schedulers. In case of EF, AF1, AF2 classes the queue length continuously is around 50 packets regardless of which scheduler is used. The reason of this

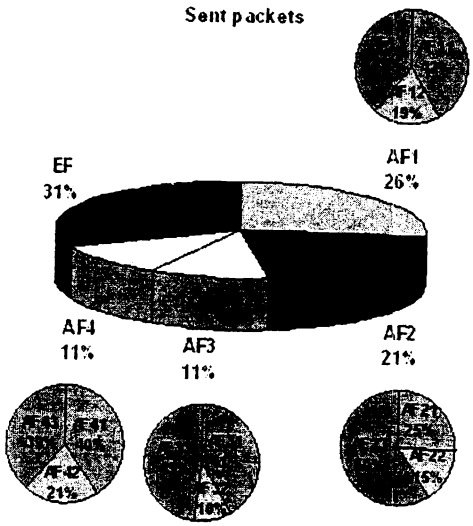


Figure 4. Sent packet ratio

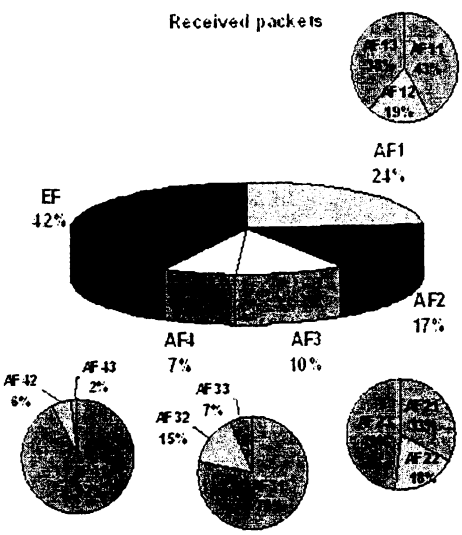


Figure 5. Received packet ratio

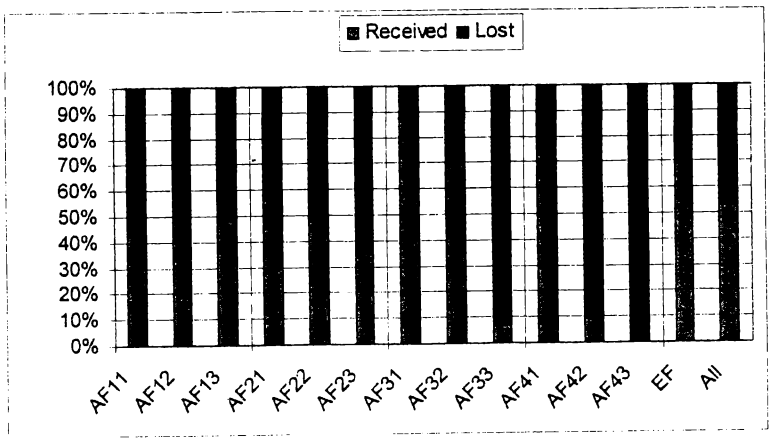


Figure 6. Packet loss statistics by classes

behavior is that we set up the RED parameters of those classes in such way that the Minth (Minimum threshold) is greater than the the link buffer capacity (50 packets). This means that there is no Early drop, only the link makes the dropping. We can observe that in case of those classes the deviation from the mean queue length is the smallest in case of PRI scheduler, and the greatest in case of WRR scheduler. WIRR is between them.

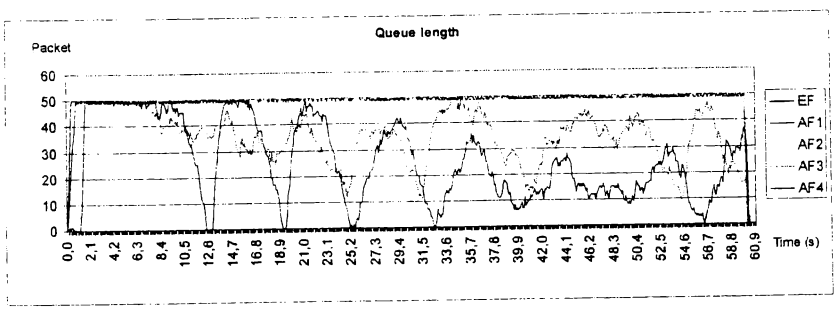


Figure 7. Queue length in case of PRI scheduler

In case of AF3, AF4 classes we can see how RED works. When the average queue size varies from Minth to Maxth packets are dropped linearly, but when the average size reaches Minth no packets are dropped. The queue length

variation (when starts a dropping, how much time it takes) also depends on the scheduler we currently use.

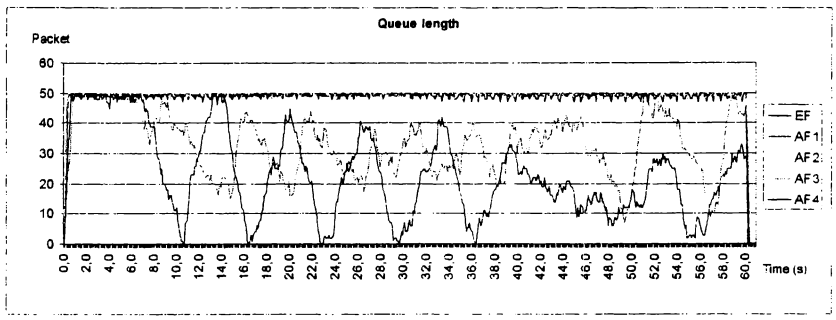


Figure 8. Queue length in case of WIRR scheduler

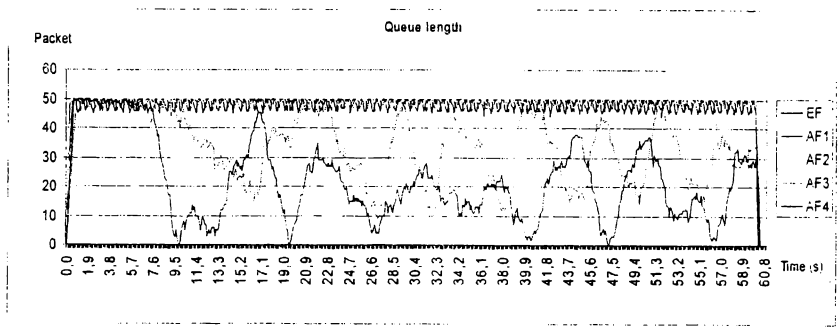
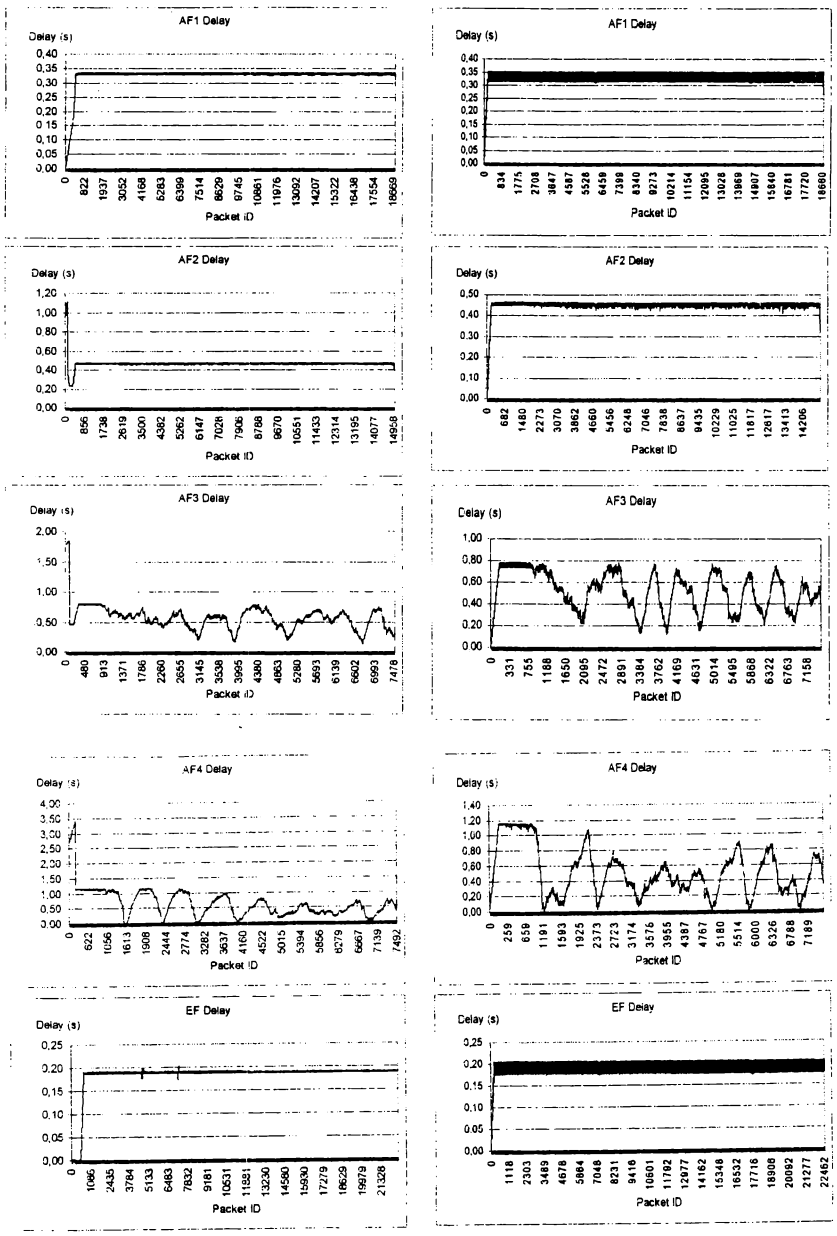


Figure 9. Queue length in case of WRR scheduler

Fig. 10 shows the delay variation of packets. We can observe that the delay varies exactly as the queue size varies. This is what we expected, the proportion of the queue size and the delay should be a constant value, which conforms to the wellknown Little-formula ($Q = \lambda * W$). Because the limited size of the article we only present the delay of packets in case of PRI and WRR scheduler, but the WIRR scheduler also holds the above criteria.

Fig. 11, 12, 13 show the throughput variation. The average realized throughput per class is the same for all schedulers. The deviation (jitter) from the mean is the smallest in case of PRI scheduler and the greatest in case of WRR, while WIRR is between them.



a) Delay in case of PRI scheduler b) Delay in case of WRR scheduler

Figure 10. Delay of packets

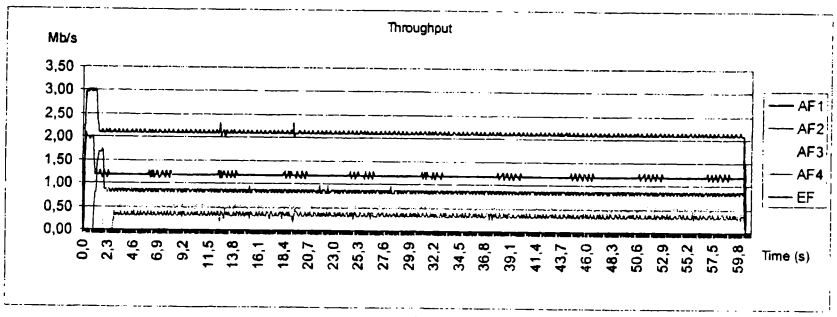


Figure 11. Throughput in case of PRI scheduler

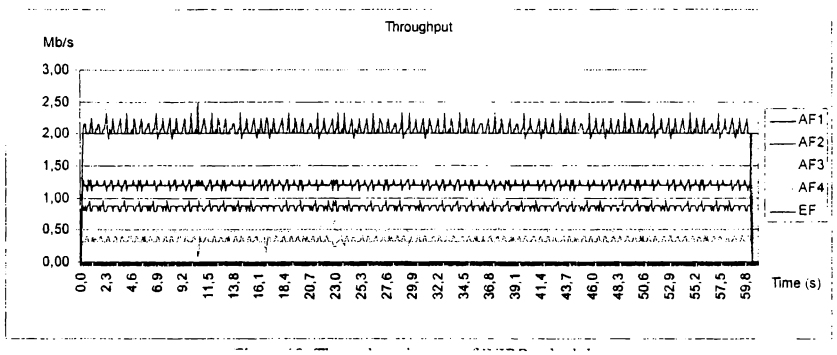


Figure 12. Throughput in case of WIRR scheduler

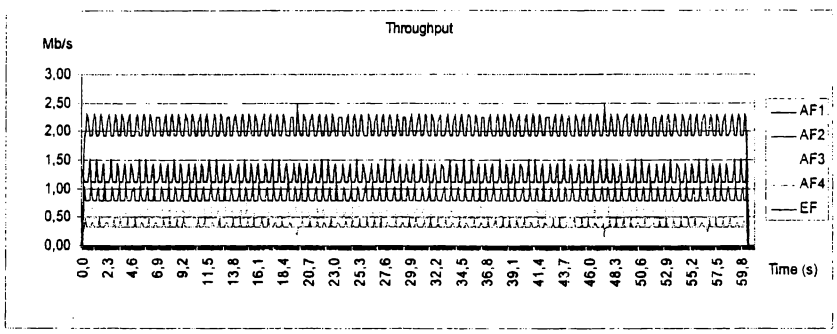


Figure 13. Throughput in case of WRR scheduler

Fig. 14 shows the arithmetic mean delay per class and a weighted (with the number of packets) arithmetic mean delay in case of three schedulers.

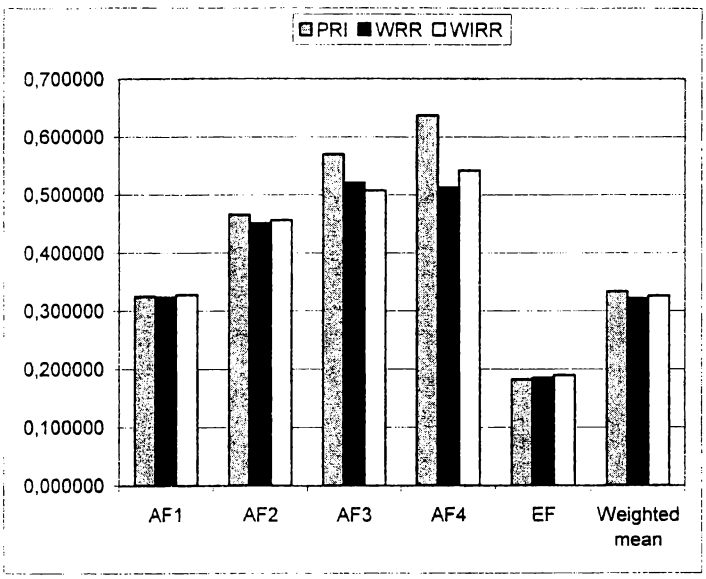


Figure 14. Arithmetic mean delays

5. Conclusions

The novelty of the paper is the comparison of the three traditional schedulers. In the earlier papers only individual schedulers were analyzed.

Acknowledgements. The authors sincerely thank to Gyula Bódog and Endre Farkas from Nokia R&D Budapest for their many helpful suggestions.

References

[1] **Blake S., Black D., Carlson M., Davies E., Wang Z. and Weiss W.,** *An architecture for differentiated services,* IETF RFC 2475, 1998.

[2] **Heinanen J., Baker F., Weiss W. and Wroclawski J.,** *Assured forwarding PHB group,* IETF RFC 2597, 1999.

[3] **Davie B., Charny A., Bennett J., Benson K., Boudec J., Courtney W. and Davari S.,** *An expedited forwarding PHB,* IETF RFC 3246, 2002.

[4] **Braden B., Clark D., Davie B., Floyd S., Jacobson V., Wroclawski J. and Zhang L.,** *Recommendations on queue management and congestion avoidance in the Internet,* IETF RFC 2309, 1998.

[5] **Floyd S. and Jacobson V.,** Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking,* **1** (4) (1993), 397-413.

[6] **Bodog Gy.,** *Comparison of packet service disciplines,* Diplomawork, Technical University of Budapest, 1999.

[7] *Tcl&Toolkit,* available via `http://www.scriptics.com`, 2002.

[8] **Fall K. and Varadhan K.,** *The ns Manual,* available via `http://www.isi.edu/nsnam/ns/ns-documentation.html`, 2002.

[9] **Chung J. and Claypool M.,** *NS by example,* available via `http://nile.wpi.edu/NS/`, 2002.

[10] **Jain R.,** *The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation and modelling,* John Wiley & Sons Inc., New York, 1991.

(Received August 21, 2004)

M. Lengyel and J. Sztrik
Dept. of Informatics Systems and Networks
University of Debrecen
H-4010 Debrecen, P.O.B. 12
lengyelm@delfin.unideb.hu
jsztrik@inf.unideb.hu

C.S. Kim
Sangji University
Wonju, Korea
dowoo@sangji.ac.kr.