

Delay Management in Core-Stateless Networks

Avadora Dumitrescu and Jarmo Harju

Abstract—Many packet scheduling mechanisms have been proposed to provide rate and delay control to flows. The mechanisms providing strict delay bounds require per-flow state and complex packet classification, which hamper their deployment on a large scale. This paper presents a scheduling mechanism for delay control that does not require per-flow state in interior network nodes (a.k.a. core-stateless networks). In our mechanism, the packet delay is un-coupled from the momentary congestion status of the node (i.e. the delay no longer depends on the length of the queue at packet arrival but rather on whether there are packets with more stringent delay requirements) and the granularity in delay differentiation becomes irrespective of the number and size of buffers. It is also shown that end-to-end per-flow delay control can be achieved through ensuring local delay management.

Index terms—quality of service, core-stateless, packet scheduling, delay management

I. INTRODUCTION

Many of the new Internet applications (e.g. voice-over-IP, videoconferencing, video streaming) have strong delay requirements on their data streams. If congestion occurs along the data path, their requirements cannot be fulfilled without service differentiation and prioritization of the packets. In this case, the end-to-end performance of a flow is strongly influenced by the choice of the packet scheduling algorithm employed in the routers.

There are several ways in which a network can provide service discrimination. One way is to provide *relative* differentiated services where the level of performance per class is given in reference with the performance level of the other classes [1]. This approach does not provide always a consistent service definition (e.g. end-to-end delay bound or throughput), because the relative quality of service (QoS) differentiation depends upon the class loads and traffic context. On the other way, *absolute* service differentiation offers strict end-to-end performance measures but imposes tight requirements on resource allocation and admission control often leading to the underutilization of network resources.

There are many packet scheduling algorithms proposed to provide rate and delay control to flows. Among the most

popular ones are *work-conserving* schedulers such as Generalized Processor Sharing (GPS) [2], [3], Weighted Fair Queuing (WFQ) [4], Earliest Deadline First (EDF) [5] and *non-work-conserving* schedulers such as Virtual Clock (VC) [6] and Core-Stateless Jitter Virtual Clock (CJVC) [7]. To support strict user requirements, WFQ and VC scheduler mechanisms provide end-to-end delays and bandwidth guarantees for users by maintaining scheduling information on a per-flow basis. CJVC is a scalable version of Jitter-VC, enabling a network to guarantee end-to-end delay similar to their core-stateful counterpart but at the expense of making the network non-work-conserving. EDF is a per-flow scheduling mechanism known to be optimal in case of a single node with single traffic envelopes and deterministic delay requirements [8]. However, for the multiple-node case, unpredictable traffic interactions make the EDF end-to-end delay guarantees questionable.

In this paper we present, analyze and evaluate a novel packet scheduling algorithm able to provide controlled end-to-end delay without specific allocation of network resources and without maintaining per-flow state in the core routers: the Core-Stateless Earliest Deadline First (CS-EDF) scheduling algorithm.

The rest of the paper is organized as follows: section II presents the proposed scheduling mechanism. Section III presents some simulation results for both the EDF and the CS-EDF approaches and compares their performances. Finally, section IV gives the conclusions and outlines future work.

II. SCHEDULING MECHANISM

The basic architecture of our approach relies on three elements: the *edge node*, the *packet state* and the *core node*, as depicted in Fig. 1.

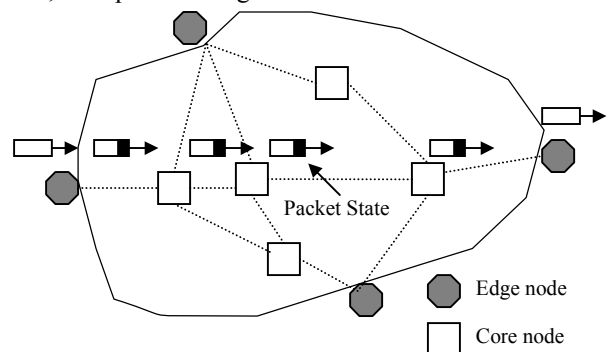


Fig. 1 Reference core-stateless network model

Avadora Dumitrescu and Jarmo Harju are with Institute of Communications Engineering, Tampere University of Technology, Finland (e-mail: avadora.dumitrescu@tut.fi, jarmo.harju@tut.fi).

The *edge* (boundary) *node* performs per-flow packet classification and traffic conditioning. The *packet state* represents a specific amount of information carried by the packet itself and updated along the network path [7], [9]. The *core* (interior) *node* does not maintain per-flow state, all the packets being processed based solely on the packet state and on information internal to the node.

The delay management information carried by the packet is:

1. The maximum remaining delay (initialized by the requesting application with the maximum acceptable delay), *maxrd*;
2. The time-stamp (initialized at the moment when the packet was emitted), *tstmp*;
3. The maximum remaining number of hops to destination, *maxrn*+1.

The delay management information carried by the packet is updated at packet arrival to a node according to:

$$\begin{aligned} \mathit{maxrd} &= \mathit{maxrd} - (\mathit{current_time} - \mathit{tstmp}) \\ \mathit{tstmp} &= \mathit{current_time} \\ \mathit{maxrn} &= \mathit{maxrn} - 1 \end{aligned}$$

i.e. *maxrd* is updated by subtracting the time elapsed since it arrived in the previous node and *tstmp* becomes the arrival time to the node. It should be noted that although it is based on the idea that the nodes are time-synchronized (i.e. their current time is the same) the updating scheme could easily be adapted for the case that no time synchronization is in place. However, without time synchronization in the network it would be impossible to keep track of the delay accumulated on the transmission lines and layer 2 devices between routers. Despite the fact that this delay may be relatively small compared with the packet processing delay we have chosen to present the variant considering synchronized clocks because it enables a consistent end-to-end delay management.

In our implementation, the packet queue in each node is organized as a linked list in increasing order of the variable:

$$\mathit{rem_del}_Q = \frac{\mathit{maxrd}}{\mathit{maxrn}} - (\mathit{current_time} - \mathit{tstmp}) \quad (1)$$

i.e. the acceptable delay for the present hop reduced by the time that the packet already spent in the queue. Taking into account that the current time is the same for all the packets in the queue the list can be linked corresponding to the increasing order of:

$$\mathit{max_leave}_Q = \frac{\mathit{maxrd}}{\mathit{maxrn}} + \mathit{tstmp} \quad (2)$$

i.e. the arrival time to the node plus the acceptable delay for the current hop. This list organization does not require any reordering of the packets once they are placed in the list. When a new packet is received it is inserted in its corresponding place in the linked list. The packets are

forwarded in the order of the linked list (i.e. in increasing order of their maximum acceptable leaving time), which means that a packet is forwarded when there are no other packets with more stringent local delay requirements. In this way the packets with not so stringent local delay requirements let the packets with more stringent delay requirements in front of them, irrespective of congestion level, even if the packets with more stringent delay requirements arrived later to the node. The mechanism has the potential drawback that malicious applications could stamp a very tight delay requirement on their packets to ensure priority. This situation can be policed at the ingress nodes by checking that the delay requirement conforms to the agreed profile as defined in the service level agreement and by setting an increasing price on tighter delay requirements.

If the packet has exceeded its maximum acceptable delay on the current hop, i.e. if

$$\frac{\mathit{maxrd}}{\mathit{maxrn}} + \mathit{tstmp} - \mathit{current_time} < 0 \quad (3)$$

then the packet could still be forwarded (when its turn comes) in the hope that the supplementary delay could be compensated on the remaining path. However, if *maxrd* becomes negative, i.e. the packet can no longer be delivered within the maximum acceptable delay (a fact that is detected upon packet arrival when updating *maxrd*), the packet could be either treated as a packet without delay requirements or be discarded (depending on the application requirements). It should be noted that the increasing possibility of reaching the negative *maxrd* situation with tighter delay requirements (and thus the increasing possibility of having more packets discarded or treated as packets without delay requirements) is a deterrent against stamping delay requirements tighter than those really needed.

The search for a packet's place in the linked list can be a simple list search starting from the list head or, in case that long lists are expected, the search can be performed on an index array or a hashing table associated with the list. It would be recommended that the management of the linked list (all operations except for forwarding the first packet in the list: list insertions, list searching including possible indexing or hashing, etc.) be a completely parallel process to the packet forwarding.

For unified treatment of all packets, the packets without a delay requirement could either be given a very large delay requirement or could simply be placed after all packets with delay requirements, in incoming order. The packet list (both for packets with and without delay requirements) could have a second link in increasing order of priorities. Thus when a packet is received under congestion conditions than mandate packet discarding, it is not the last arrived packet that is discarded but rather the packet with the smallest priority (first packet in the second link-ordering).

III. SIMULATION

A. Simulation scenario

In order to assess the scheduling mechanism operation we have simulated the packet transfer over the same path for four flows (A, B, C and D) with different end-to-end delay requirements. The delay requirements were: 40, 60 and 100 ms for A, B and C. No delay requirement was specified for the packets of the D flow. The flows, referred to as UDP flows in the rest of the paper, have fixed packet length (1024 bits) and interarrival times uniformly distributed between $0.5 \cdot avgint$ and $1.5 \cdot avgint$, with $avgint$ being the average interarrival time derived from the bitrate. The bitrates were: 0.3, 0.2, 0.2 and 0.1 Mb/s for the A, B, C and D flows, respectively.

Two ON-OFF cross traffic flows (E and F) were transferred through the internal nodes of the path in order to provide a variable congestion level both along the path and in time. The cross traffic flows' bitrates were 0.2 Mb/s on the average, with 0.6 Mb/s in the ON periods. All the links on the path had a capacity of 1Mb/s and the transmission delay between nodes was taken to be 2 ms. Fig. 2 gives the transfer path configuration for all the flows included in the simulation.

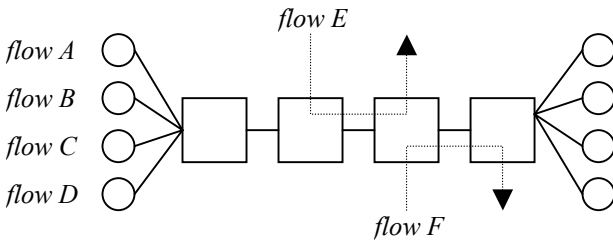


Fig. 2 Configuration of the transfer path used in the simulation

A delay requirement of 8 ms was chosen for the cross traffic flows to induce competition in packet treatment.

B. Simulation results for EDF

In order to have a comparison reference for the performance of our core-stateless delay management approach we have also performed the simulation using the classical EDF scheduling. The EDF scheduling associates to each flow i at each traversed node/router n a local delay bound d_i^n . An incoming packet belonging to flow i is stamped with a departure deadline $t+d_i^n$ (where t is the arrival time) and the packets are forwarded in the increasing order of their deadlines.

It has been shown that, in a deterministic setting, EDF is the optimal scheduling policy at a single node [8], [11]. In a multi-node path with crossing traffic flows, however, the interactions that distort the flow envelopes bounding the traffic are no longer deterministic and the EDF

behavior becomes suboptimal [12]. Fig. 3 shows the end-to-end delay provided by the EDF scheduling mechanism for the flows A and C.

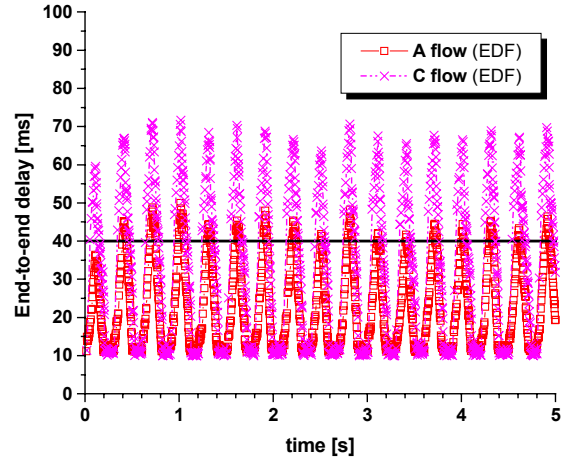


Fig. 3 EDF end-to-end delay for the A and C flows

It can be seen that, while the 100 ms delay bound is respected for flow C, the much stricter 40 ms delay bound for flow A is violated systematically during the congestion periods when both the crossing traffic flows are ON. Over the 5 s period shown in Fig. 3, 174 from the 1532 packets of flow A were delayed with more than 40 ms along the path.

Also the delay bound for the B flow was violated due to the strong competition for the earliest departure deadline during the ON periods of the ‘crossing traffic’, as can be noticed in Fig. 4. Here, owing to the more relaxed delay requirements, only 17 out of 1042 packets in the analyzed 5 s period got delays above the 60 ms requirement.

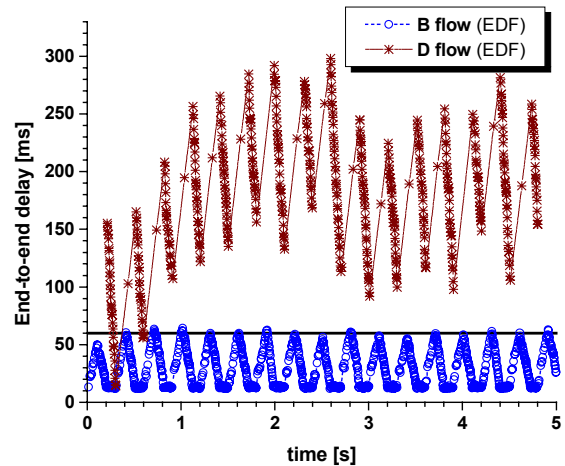


Fig. 4 EDF end-to-end delay for the B and D flows

C. Simulation results for CS-EDF

Fig. 5 presents the end-to-end delay provided for the A and C flows by our CS-EDF scheduling mechanism.

A clear delay differentiation under delay-competition conditions can be observed. All packets are within the imposed delay bounds – the points where the A flow packet delays seem to be above the 40 ms line are owed to the large symbols used for visibility. The maximum delay experienced by flow A packets over the simulated transmission path was 39.88 ms.

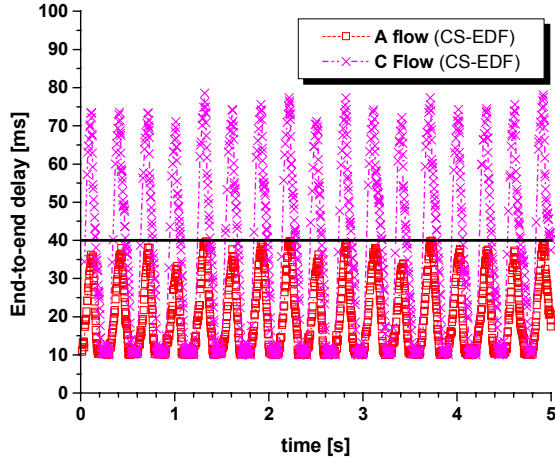


Fig. 5 CS-EDF end-to-end delay for the A and C flows

Also in the case of flow B the end-to-end delay bound was satisfied for all packets, as can be seen in Fig. 6. Again, a clear delay differentiation can be observed between flows sharing the same transfer-path context.

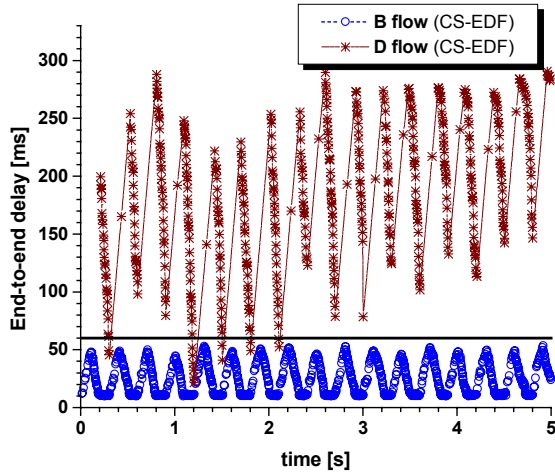


Fig. 6 CS-EDF end-to-end delay for the B and D flows

D. Comparison between EDF and CS-EDF

It should be noted that, because EDF does not take into consideration the transmission delay between nodes, a preemptive approach for calculating the per-node delay bounds was used. Instead of generating the EDF delay bound per-node from the end-to-end delay, the per-node delay requirement was obtained using the difference between the end-to-end delay bound and the total

transmission delay. For example, in the case of flow A the EDF delay bound per node was $(40-3*2)/4=8.5$ ms. In this way it was ensured that the synchronized-clock CS-EDF does not have an unfair advantage from the fact that it takes into consideration the transmission delay.

When comparing the EDF and CS-EDF performance within our simulation framework, it is interesting to note that the averaged end-to-end packet delays do not show a significant or systematic difference for any of the flows. Despite the packets that violate the delay bounds, the per-flow averaged delays in EDF were well within the delay bounds. However, there are subtle yet crucial differences between EDF and CS-EDF when looking at the delay variations between packets and between nodes.

Mechanisms like EDF provide differentiation and optimality at a single node, but do not provide dynamic adaptation if the packet violates its local deadline. The improved delay management capability of CS-EDF comes mostly from the ability to discriminate the delay treatment between the packets of the same or different flows depending on their different delay history. For example, in CS-EDF, the packets that have been delayed more in node 2, due to the ON state of cross-traffic from flow E, have tighter delay requirements on the remaining path (particularly in node 3) and get a faster transfer (occasionally passing in front of F-flow packets). In the meantime, in EDF, the packets that have experienced higher delay in node 2 can recover the lost grounds only if the traffic context on the remainder of the path enables a faster-than-required transfer. Thus, once the local delay bounds are violated, there is no mechanism in the EDF approach to pursue purposely the recovery of the excess delay. In CS-EDF, once a local delay violation occurs, there is a systematic attempt to compensate it along the remaining path. Similarly, if the packet is ahead with respect to its local delay bound, it receives a larger delay requirement on the downstream path, allowing packets with more stringent delay requirements to be forwarded in front of it.

CS-EDF also promotes a reduced jitter since packets that have experienced a local delay violation upstream obtain tighter delay requirements downstream while packets that have experienced a fast forwarding treatment upstream obtain larger delay requirements downstream. CS-EDF attempts to ensure the end-to-end delay requirement as accurately as the network context permits and attempts to dynamically compensate the delay variations along the packet transfer path.

It should be noted that, in case of a nondeterministic rapidly fluctuating network context, any delay management measures taken at the flow level are less effective than similar measures taken at the packet level.

IV. CONCLUSIONS

A novel scheduling mechanism, the Core-Stateless Earliest Deadline First (CS-EDF) scheduling algorithm, was presented. CS-EDF provides controlled end-to-end delay without specific allocation of network resources and without maintaining per-flow state in the core routers. CS-EDF ensures that the local packet delay is un-coupled from the momentary congestion status of the node (i.e. the delay no longer depends on the length of the queue at packet arrival but rather on whether there are packets with more stringent delay requirements). Also the granularity in delay differentiation becomes independent on the choice of buffer number and length (the delay differentiation granularity depends only on the granularity in representing $maxrd$). Furthermore, the use of a double-linked list, with separate ordering for forwarding priority and for delay management, ensures isolation between the packet acceptance decision and the delay control.

Under dynamically variable conditions, both in time and along the packet transfer path, CS-EDF intrinsically attempts to compensate the local delay variations (from the local delay requirements) along the remaining path. Thus, local delay violations can be compensated even under adverse network conditions and the delay jitter is reduced as much as the network context permits. All these are done with a packet-level granularity, which has been attempted by very few other delay management mechanisms.

The simulations have clearly shown that CS-EDF ensures delay discrimination under delay competition conditions. Also it was shown that under adverse network conditions CS-EDF is capable of providing the delay guarantees when the classical stateful EDF is no longer able to assure the delay guarantees.

A large amount of further tests are required for a systematic assessment of CS-EDF delay management capabilities. First of all, more complex simulations have to be carried out using a realistically complex network and traffic context. Also the delay management performance should be tested in conjunction with admission control mechanisms and against real application requirements. A much greater number of sources, with a wider range of behaviors and requirements should be used to analyze the delay treatment differentiation and its granularity. Also more systematic comparisons with other delay management schemes should be conducted. Finally a mathematical analysis of the computational complexity, achievable bounds and limits should be pursued to evaluate CS-EDF's optimality.

REFERENCES

- [1] C.Dovrolis and P. Ramanathan, "A case for relative differentiated services and the proportional differentiation model", IEEE Network, vol. 13, pp. 26-34, Sept./Oct. 1999.
- [2] A. Parekh and R.Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case", IEEE/ACM Transactions on Networking, vol. 1(3), pp. 344-357, June 1993.
- [3] A.Parekh and R.Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case", IEEE/ACM Transactions on Networking, vol. 2(2), pp. 137-150, April 1994
- [4] A.Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm", Proceedings of ACM SIGCOMM, pp. 3-12, 1989.
- [5] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks", IEEE Journal of Selected Areas in Communications, vol. 8(3), pp. 368-379, April 1990.
- [6] L.Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks", Proceedings of ACM SIGCOMM, pp.19-29, September 1990.
- [7] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management", Proceedings of ACM SIGCOMM 1999, pp. 81-94, August 1999.
- [8] L. Georgiadis, R. Guerin, and A. Parekh, "Optimal multiplexing on a single link: Delay and buffer requirements", IEEE Transactions on Information Theory, vol. 43(5), pp.1518-1535, Sept. 1997.
- [9] I. Stoica, S. Shenker, and H.Zhang, "Core-Stateless Fair Queueing : A Scalable Architecture to Approximate Fair Bandwidth Allocations in High-Speed Networks", IEEE/ACM Transactions on Networking, vol. 11(1), February 2003.
- [10] Z.I. Zhang, Z. Duan and Y.T. Hou, "Virtual Time Reference System: A Unifying Scheduling Framework for Scalable Support of Guaranteed Services", IEEE Journal on Selected Areas in Communications, vol. 18(12), December 2000.
- [11] J. Liebeherr, D. Wrege and D. Ferrari, "Exact admission control for networks with a bounded delay service", IEEE/ACM Transactions on Networking, vol. 4(6), pp. 885-901, Dec. 1996.
- [12] H. Zhang and D. Ferrari, "Rate -controlled service disciplines", IEEE Journal on Selected Areas in Communications, 13(6), pp.1071-1080, August 1995.