Proceedings of the IEEE Conference on High Performance Switching and Routing (ATM 2000), Heidelberg, June 2000, pp. 357-364.

A Scheduling Algorithm for Relative Delay Differentiation

Stefan Bodamer, University of Stuttgart, Institute of Communication Networks and Computer Engineering, Pfaffenwaldring 47, 70569 Stuttgart, Germany

Abstract

An important issue in the design of multi-service IP networks is to find appropriate mechanisms to provide different levels of quality of service (QoS) for different types of applications. In this paper, scheduling mechanisms for real-time traffic in a differentiated services IP network are studied. A new scheduling algorithm called *weighted earliest due date (WEDD)* is proposed that provides tunable delay differentiation for real-time applications like voice over IP. WEDD is an enhancement of the earliest due date (EDD) scheduler in the sense that not only different deadlines but also different deadline violation probabilities are provided. The ratios of the violation probabilities are directly defined by a set of parameters. Simulation studies are performed to show that the scheduler is able to maintain the specified ratios under various load conditions.

1 Introduction

The requirements of many applications and users are no longer satisfied by the best-effort service of the current Internet. Therefore a lot of research has been done to find an architecture that provides quality of service (QoS) in an appropriate manner. Two main streams can be distinguished: reservation and prioritization.

Reservation of resources on a per-flow basis is used to give deterministic or statistical guarantees related to certain QoS measures. In the Internet community, this solution is represented by the integrated services (intserv) architecture using RSVP as a signalling protocol [3]. Similar traffic control functions are necessary as in ATM networks.

In a pure prioritization approach only relative guarantees are given. One can expect that flows belonging to a higher priority class receive better performance than those of a lower priority class (either in a deterministic or statistical sense) but absolute values for the performance parameters are not previously known. The differentiated services (diffserv) architecture defined in [2, 16] provides a framework for relative differentiation of traffic classes which are identified based on a set of bits in the IP packet header.

An appealing feature of diffserv is that it handles flow aggregates thus avoiding per-flow state in the network. There is no end-to-end signalling of traffic parameters and QoS requirements at flow setup. Admission control is not required if only relative differentiation is provided. The realization of a diffserv network requires the definition of per hop behaviours (PHB) implemented in the network nodes. The assured forwarding (AF) PHB group [10] comprising four AF classes with three drop precedence levels each and the expedited forwarding (EF) PHB [11] have already been specified.

An important issue in a diffserv network is to define and manage PHBs so that an appropriate differentiation is possible for all types of traffic. Non real-time (elastic) traffic (e.g. WWW, file transfer, e-mail) is usually based on TCP which enables applications to adapt their sending rate in case of congestion. This type of traffic needs a differentiation with regard to throughput. Real-time traffic produced by applications like voice over IP or video conferencing additionally requires delay differentiation. As the requirements of elastic and real-time traffic are different the basic assumption in this paper is that both types are separated on each output link within a diffserv node. Such a separation is not unusual. The diffserv implementation SIMA (simple integrated media access), e.g., follows the same principle [14, 18].

The separation of real-time and non real-time traffic leads to a hierarchical resource sharing architecture (**Fig. 1**) which partly coincides with the AF approach. Each type gets a share of the link bandwidth C (C_{rt} and C_{nrt}, respectively) that can be adapted dependent on the current load. Link sharing is realized by a fair queueing scheduler, e.g. weighted round robin (WRR) [12], self-clocked fair queueing (SCFQ) [9], or class-based queueing (CBQ) [8]. Within the non real-time traffic category an active buffer management strategy

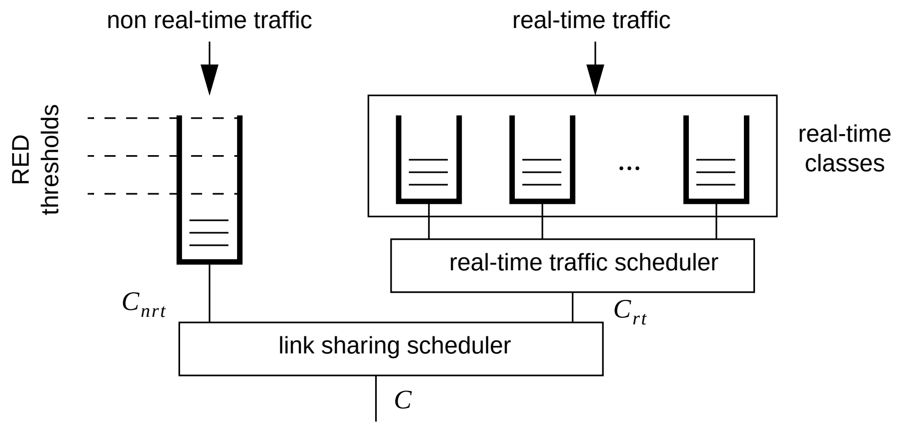

Fig. 1 Hierarchical resource sharing architecture

with different congestion level thresholds based on random early detection (RED) [7] can be employed. For real-time traffic, however, buffer management is not sufficient to provide differentiation. Delay differentiation is better realized by a scheduling mechanism distinguishing several real-time classes.

The rest of the paper concentrates on finding a scheduling mechanism for the real-time classes that provides relative delay differentiation in an appropriate manner. Section 2 discusses several existing alternatives. In Section 3, a new scheduler is proposed that assigns different deadlines to packets in different real-time classes and enforces weighted deadline violation probabilities. The performance of the scheduler is evaluated in Section 4 using simulations.

2 Scheduling Algorithms for Delay Differentiation

2.1 Static Priority Queueing

A simple form of providing delay differentiation is to assign static delay priorities to different classes. In this scheme, packets in class i are not forwarded until all queues with priority index greater than i are empty.

The static priority model, however, has several serious drawbacks. First, under heavy load conditions starvation may occur in lower priorities. Moreover, the grade of differentiation between classes extremely depends on the load and the load distribution. Finally, a static priority scheme is not controllable as it misses any tuning knobs which could help the administrator to configure the system.

2.2 Fair Queueing

A different approach, which could be easily integrated into the architecture shown in **Fig. 1**, is to use a fair queueing mechanism also for real-time traffic. Increasing the weight of a class generally results in better performance with respect to delay. However, it is a complex task to find appropriate values for the weights even in an ideal generalized processor sharing (GPS) scheduler [21]. The differentiation has to be significant on the one hand and the system should not degenerate to a static priority multiplexer on the other hand. This is a problem especially since the distribution of load on different classes, which is previously unknown in a diffserv network, has a huge impact.

A solution could be an adaptation of the weights based on online traffic measurements. A long term adaptation, however, may result in lower performance of high priority classes in short time scales [6]. On the other hand, if the adaptation interval is small, the algorithm has not much in common with fair queueing any longer.

2.3 Mean Delay Proportional Schedulers

Recently, several schedulers have been proposed that provide relative differentiation with respect to mean delays [4, 6, 15]. In [4, 6] the well-known waiting time priority (WTP) scheduler is shown to give a rather exact proportional differentiation of mean delays in both short and long time scales.

A disadvantage of this kind of schedulers is that they are designed to provide delay differentiation based on average delays. The performance of multimedia tools like IP telephony applications will not so much depend on mean delays but on the probability that the transmission delay exceeds a certain threshold [1]. A WTP scheduler also differentiates traffic with regard to the excess probabilities. But the ratio of the excess probabilities obtained for different classes depends on the load distribution and thus cannot be fixed by simply adjusting the scheduling parameters.

2.4 Earliest Due Date (EDD) Scheduler

The EDD scheduler [13], also denoted as EDF (earliest deadline first), is a means to provide absolute delay differentiation [5]. Each class i is associated with a delay bound δ_i. A packet of class i arriving at time t_A receives a tag $t_A + \delta_i$ representing its deadline. The packets to be forwarded are scheduled in increasing order of their deadlines.

The EDD scheduler has usually been applied in conjunction with per-flow reservation. In that case the number of flows is restricted by admission control in order to guarantee that the delay bounds are not exceeded absolutely or at least with a certain probability. For an analytic evaluation of the EDD deadline violation probabilities in this context see [19].

If EDD scheduling is applied in a diffserv network without admission control the property becomes important that in a homogeneous traffic scenario the probabilities of deadline violations due to congestion are equal in all classes. This property holds independent of the total load and the load proportion of the different classes.

An additional option in operating an EDD scheduler is to discard packets that exceed their deadline before they enter service. This leads to a scheduling mechanism called shortest time to extinction (STE) in [17]. The STE policy is especially appropriate for handling real-time packets which become useless if they do not reach their destination within a certain time interval.

3 Proposed Scheduler: WEDD

In this section, a new scheduler called weighted earliest due date (WEDD) is proposed. WEDD is an enhancement to EDD in the sense that it not only provides different deadlines but also different deadline violation probabilities. The violation probabilities p_i are weighted according to given weight parameters w_i such that

$$\frac{p_i}{p_j} = \frac{w_i}{w_j} \qquad (1)$$

This generally results in better performance for a class with lower weight parameter.

First, this gives better controllability to the administrator as only the weight parameters have to be set. Moreover, such a scheme is useful if there is a direct relationship between the delay bounds specified in the scheduler and the playout delays in real-time applications like voice over IP. In this case packets exceeding their deadlines have to be discarded in the end device leading to a reduction of audio or video quality. So the proposed scheduler gives the ability to directly and separately influence delay and loss which are perceived by the user in different ways.

The basic operation of WEDD, i.e. setting deadlines $t_A + \delta_i$ on arrival at time t_A and scheduling packets in increasing order of their deadlines, is the same as for EDD. However, operation mode changes if congestion occurs in more than one class. More exactly, a class i is called "congested" if the first packet that is backlogged in that class has a deadline $t_A + \delta_i < t_S + \Delta_i$ where t_S denotes the current system time and Δ_i is a safety margin, e.g. $\Delta_i = \delta_i / 10$. This means the first packet in class i is about to violate its deadline. In this case system operation switches to "congestion mode".

In congestion mode, for each congested class a congestion tag c_i is calculated:

$$c_i = \frac{w_i}{E_i(t_S)} \qquad (2)$$

In this equation, $E_i(t_S)$ is a measurement-based estimation of the real deadline violation ratio of class i at current time t_S. After the congestion tags for all congested classes have been determined the packet with the lowest congestion tag c_i is served.

There are different ways to measure the real deadline violation ratio. The simplest approach is to have two counters m_i and n_i in each class that count the aggregate length of packets having exceeded their deadlines (m_i) and the total aggregate packet length in that class (n_i). The estimated deadline violation ratio is then simply given by the ratio of the current values of m_i and n_i. To avoid that the counter values become infinitely large the value of both m_i and n_i can be multiplied by some factor $\alpha \in [0, 1]$ each time n_i is updated. For $\alpha < 1$ a limitation of m_i and n_i can thus be achieved. Losses that occurred in the past are considered with an exponentially decreasing weight in that case.

Like EDD, WEDD can be specified to discard late packets. More exactly, the scheduler checks whether the deadline of the first packet in each class is smaller than the current system time before considering this packet for determination of the smallest tag. In this case the packet is deleted and the next packet in this class is checked in the same way. When the discarding option is applied the WEDD scheduler becomes similar to the one presented in [20]. The authors in [20], however, assume that packets are discarded on arrival.

Furthermore, their scheduling and discarding scheme aims at providing statistical guarantees as necessary, e.g., in an ATM environment. In the context of this paper, absolute violation probabilities are unknown since there is no admission control and the focus is on relative differentiation.

The WEDD scheduler provides appropriate tuning knobs in form of delay bounds and weight parameters. One may argue that this property is purchased by an increased implementation complexity as, e.g., a two-stage tagging of packets and a counter-based estimation of deadline violation ratios is required. It has to be considered, however, that only a rather small number of real-time classes (e.g., 4, 8, 16, or 32) will be defined. Therefore the scalability problem, which is known to be a killer argument in a network with per-flow reservation, does not occur in this context.

Another issue in connection with network integration is end-to-end delay. Any scheduler among those previously presented can only influence performance parameters related to a single node. The quality perceived by a real-time application user, however, depends on loss and delay occurring on the total end-to-end path. Probably, in most cases congestion will occur only in a single node on the end-to-end path. If there is congestion in more than one node on the end-to-end path additional mechanisms could help to limit end-to-end delay. One mechanism could be to reassign a packet that has been delayed in a first congested node to a different real-time class with tighter delay bound.

i	δ_i	w_i
0	16 ms	1
1	32 ms	1
2	64 ms	1
3	128 ms	1
4	16 ms	10
5	32 ms	10
6	64 ms	10
7	128 ms	10

Table 1 Sample values for delay bounds and weight parameters

An example may help to clarify this approach. Let us assume that there are 4 different delay bounds and 2 different values for the weight parameters yielding a total number of 8 real-time classes (**Table 1**). If we regard a packet belonging to class 3 (delay bound 128 ms) that is delayed by, e.g., a value of 60 ms in a first congested node it would be appropriate to reassign that packet to class 2. If the modified delay bound of 64 ms can be hold in a subsequent congested node it is also guaranteed that the original delay bound of 128 ms is not exceeded. Of course, a reassignment of packets has to be carefully considered in order to avoid a dramatic increase of high priority (i.e. low delay bound) traffic.

4 Performance Evaluation

This section concentrates on the evaluation of real-time scheduling taking only the right side of **Fig. 1** into account. Non real-time traffic is assumed to permanently consume its complete bandwidth share C_{nrt}. This is a quite realistic assumption for a congestion situation due to the greedy behaviour of TCP-based sources. The consequence for the real-time traffic part is that there is never more than C_{rt} available for this kind of traffic. Thus, the real-time traffic part can be modelled as a single server with capacity C_{rt}.

For real-time traffic modelling a simple burst traffic model is considered where bursts arrive according to a Poisson process. The number of packets in a burst follows a geometric distribution with mean $b = 40$. During a burst packets of constant length $L = 200$ Bytes arrive in fixed time intervals. The arrival rate h within a burst is set to 200 kbit/s, which is 1/50 of the link capacity $C_{rt} = 10$ Mbit/s reserved for real-time traffic. The burst arrival rates λ_i in the different classes as well as the total burst arrival rate $\lambda = \Sigma\lambda_i$ are varied during the following simulations. The WEDD estimator parameter α is set to 1 in all cases.

A first scenario comprises two real-time classes ($i \in \{0, 1\}$) with delay bounds $\delta_0 = 100$ ms, $\delta_1 = 50$ ms, a weight parameter ratio $w_0/w_1 = 10$, and congestion margins $\Delta_0 = \Delta_1 = 10$ ms. The total burst arrival rate λ is fixed such that the offered load $\rho = \lambda \cdot L \cdot b/C_{rt}$ equals 95%. The ratio of burst arrival rates λ_1/λ_0 is varied in a range between 0.1 and 10.

The simulation results (including 95% confidence intervals) of the deadline violation probabilities for EDD and WEDD in a system that discards late packets are depicted in **Fig. 2**. It is obvious that over the complete range of the load distribution WEDD is able to provide the desired ratio of violation probabilities very exactly. EDD on the other hand yields equal violation probabilities in both classes as expected. If packets that have exceeded their deadlines are not discarded the deadline violation probabilities increase by about one order of magnitude (**Fig. 3**). In this case the differentiation by the desired factor of 10 is not possible when the load share of class 1 traffic is very high. An explanation is that due to the extreme load distribution

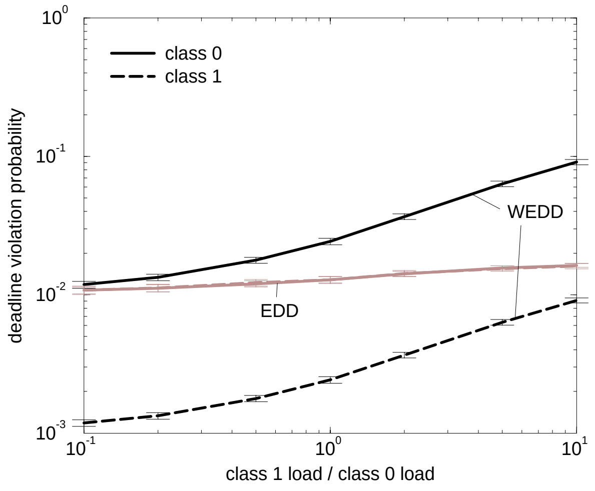

Fig. 2 EDD and WEDD with discarding, 2 classes with constant total load 95%

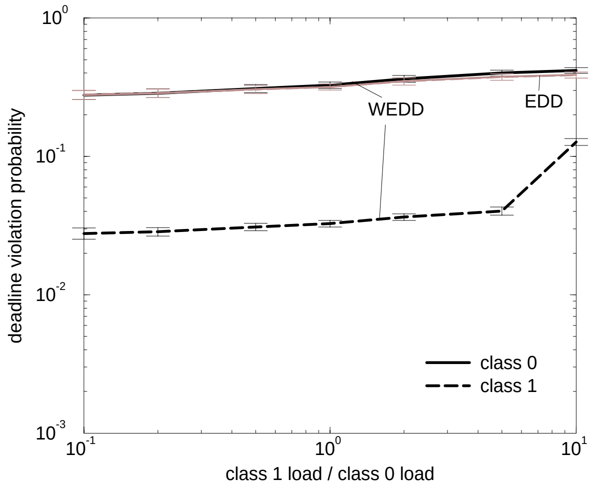

Fig. 3 EDD and WEDD without discarding, 2 classes with constant total load 95%

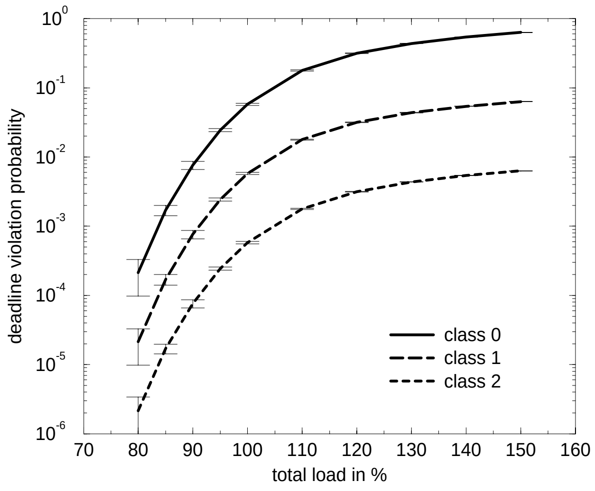

Fig. 4 WEDD with discarding, 3 classes with constant load shares (50%, 25%, 25%)

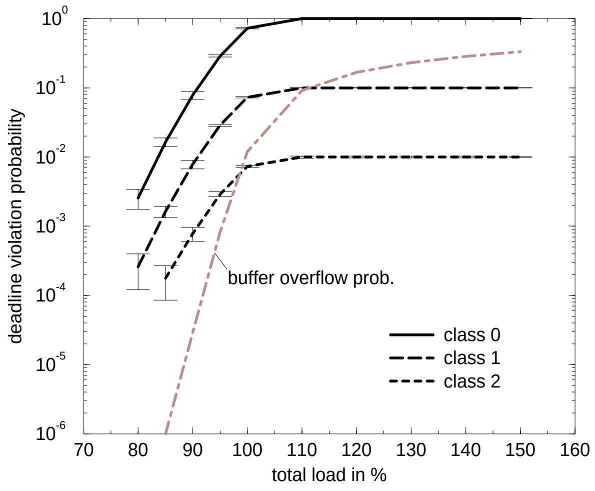

Fig. 5 WEDD without discarding, 3 classes with constant load shares (50%, 25%, 25%)

there are not enough class 0 packets to delay in favour of class 1 packets if congestion occurs. Even in the case of a static priority model, which represents the most extreme form of differentiation, it would not be possible to obtain a delay differentiation by a factor of 10 if the offered load caused by high priority packets approaches 100%. So the deviation from the desired ratio of deadline violation probabilities as obvious in **Fig. 3** is not a specific problem of the WEDD scheduler but a basic problem associated with the specific load situation.

In a second scenario, a system with three real-time classes ($i \in \{0, 1, 2\}$) is evaluated. The parameters are: $\delta_0 = \delta_1 = 100$ ms, $\delta_2 = 50$ ms, $\Delta_0 = \Delta_1 = \Delta_2 =$ 10 ms, $w_0 \div w_1 \div w_2 = 100 \div 10 \div 1$. The load shares of classes 0, 1, and 2 are fixed to values 50%, 25%, and 25%, respectively. Discarding of late packets is switched on.

In **Fig. 4**, the deadline violation (i.e. discarding) probabilities under variable total offered load ranging from modest congestion (80%) to heavy overload (150%) are shown. The ratios of violation probabilities follow the specified values of the weight parameters in lower as well as in high load regions. Note that even for a load greater than 100% no additional mechanism to limit the queue size is required (if there is a reasonable amount of buffer, see below) as packets exceeding their deadlines are discarded.

If discarding of late packets is not applied buffer limitation becomes more relevant[1]. The desired differentiation of deadline violation probabilities is achieved over the whole range of load also in this case (**Fig. 5**). However, the loss probability due to buffer overflow,

1. Buffer is assumed to be completely shared between real-time classes.

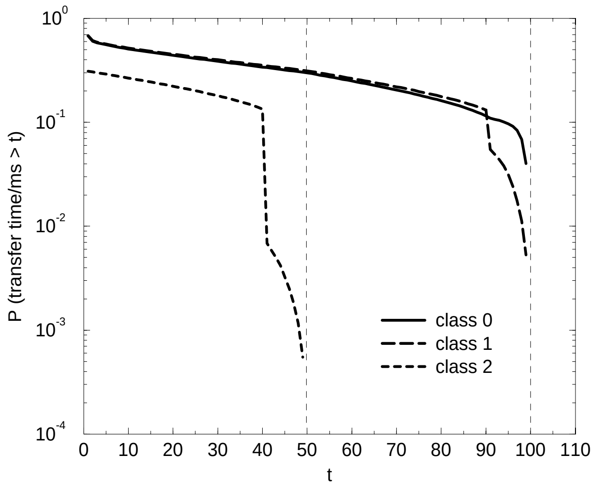

Fig. 6 Transfer time ccdf for WEDD with discarding, total load 95%

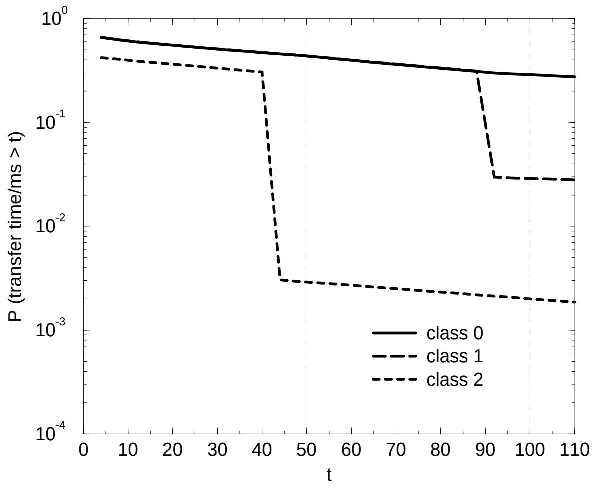

Fig. 7 Transfer time ccdf for WEDD without discarding, total load 95%

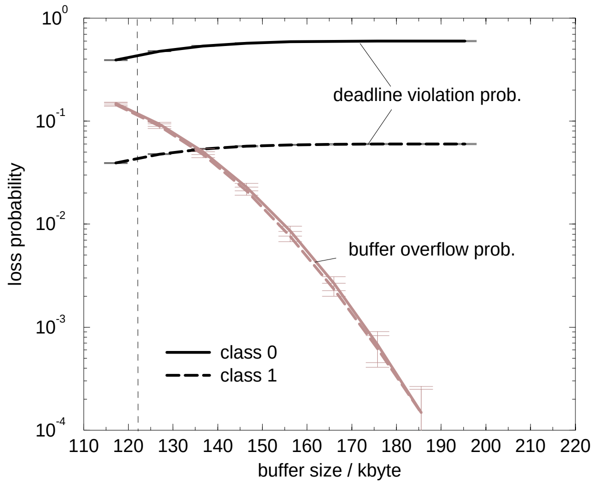

Fig. 8 Loss probability due to deadline violation and buffer overflow, 2 classes, load 150%

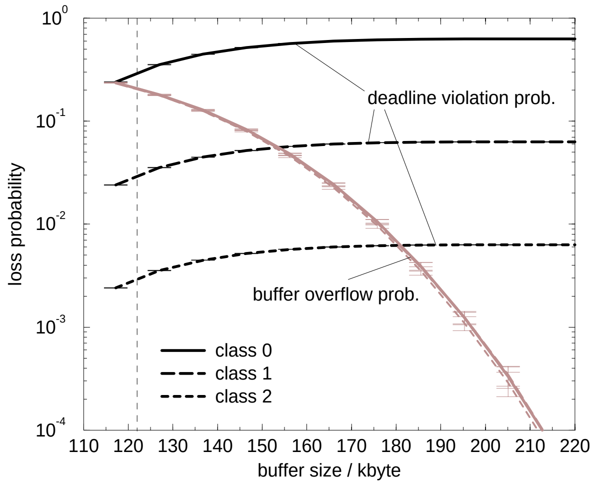

Fig. 9 Loss probability due to deadline violation and buffer overflow, 3 classes, load 150%

which is the same for all real-time classes, becomes more and more dominant as the load exceeds 100%.

Fig. 6 and **Fig. 7** give more insight into what happens to packets that are successfully transmitted. The complementary cumulative distribution functions (ccdf) of the transfer time have a sharp knee at time value $\delta_i - \Delta_i$ for the higher priority classes 1 and 2. This means that a significant amount of packets are transmitted “just in time”, i.e. they have transfer times close to their delay bounds. In the non discarding case (**Fig. 7**) the curves are continued with equal slope. Moreover, it can be observed that the behaviour for time values smaller than $\delta_i - \Delta_i$ is the same for classes 0 and 1, which have equal delay bounds. One can conclude that the ccdf behaviour in that region only depends on the value of the delay bound and is independent of the weight parameters.

The observation that buffer limitation may prevent proper differentiation if discarding of late packets is not applied gives rise to the question whether buffer size can also have an influence if late packet discarding is switched on. Therefore additional simulations with varying buffer size have been performed for the two and the three classes scenarios. For both cases a very high offered load of 150% has been chosen to produce an especially bad case. In the case of two real-time classes a buffer size of less than 1.5 times the buffer space corresponding to the 100 ms delay bound (vertical line) is sufficient to guarantee that the loss probability due to buffer overflow on packet arrival is negligible as compared to the deadline violation probabilities (**Fig. 8**).

In the three classes scenario the buffer overflow probability is higher (**Fig. 9**). Moreover, the deadline violation probability in the highest priority class is smaller as compared to the two classes scenario. Therefore a larger buffer is needed to avoid a negative influence of buffer overflow. A factor of 2 as compared to the buffer space related to the 100 ms delay bound should be far enough in this case. Note however, that in more realistic load situations a significantly smaller amount of buffer would be required.

An alternative solution would be the implementation of an additional buffer management mechanism where packets of class i are only admitted to the buffer if the buffer content does not exceed a class specific threshold θ_i. An evaluation of such a mechanism, however, is beyond the scope of this paper.

5 Conclusions

A new scheduling algorithm called weighted earliest due date (WEDD) has been proposed which provides relative delay differentiation for real-time traffic in a diffserv network. WEDD enhances the well-known EDD service discipline such that the ratio of deadline violation probabilities can be fixed to certain values. This mechanism can be combined with the discarding of late packets. So real-time applications like voice over IP can be supported in a flexible way as delay and loss, which are perceived by the user in different ways, can be influenced separately.

Simulation results have shown that WEDD is able to maintain the desired ratios of deadline violation probabilities under various traffic load conditions. The additional option of discarding late packets, which would be discarded in the receiver anyway, in the network node improves performance significantly. Moreover, this option makes it possible to do without an additional buffer management controlling packets at buffer entry. If the buffer size is well dimensioned the desired properties are achieved even in the case of very high offered load.

Further work will be necessary to evaluate the short-term behaviour of the WEDD scheduler and to perform an integrated investigation of the diffserv node including non real-time traffic management. Future evaluations will also include more “realistic” traffic characteristics based on measurements in addition to the synthetic traffic model used in this paper.

Acknowledgement

I would like to thank my colleagues Klaus Dolzer, Martin Lorang, Rolf Sigle, and Jan Späth for fruitful discussions and valuable comments on the paper.

References

[1] C. M. Aras, J. Kurose, D. Reeves, H. Schulzrinne: “Real-Time Communication in Packet-Switched Networks.” *Proceedings of the IEEE*, Vol. 82, No. 1, Jan. 1994, pp 122-139.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss: *An Architecture for Differentiated Services*, IETF, RFC 2475, Dec. 1998.

[3] R. Braden, D. Clark, S. Shenker: *Integrated Services in the Internet Architecture: an Overview*, IETF, RFC 1633, July 1994.

[4] C. Dovrolis, P. Ramanathan: “A Case for Relative Differentiated Services and the Proportional Differentiation Model.” *IEEE Network*, Vol. 13, No. 5, Sep.-Oct. 1999, pp. 26-34.

[5] C. Dovrolis, D. Stiliadis: “Relative Differentiated Services in the Internet: Issues and Mechanisms.” *Proceedings of ACM SIGMETRICS ’99*, Atlanta, May 1999.

[6] C. Dovrolis, D. Stiliadis, P. Ramanathan: “Proportional Differentiated Services: Delay Differentiation and Packet Scheduling.” *Proceedings of ACM SIGCOMM ’99*, Cambridge, USA, Sep. 1999.

[7] S. Floyd, V. Jacobsen: “Random Early Detection Gateways for Congestion Avoidance.” *IEEE/ ACM Transactions on Networking*, Vol. 1, No. 4, Aug. 1993, pp. 397-413.

[8] S. Floyd, V. Jacobsen: “Link-sharing and Resource Management Models for Packet Networks.” *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, Aug. 1995, pp. 365-386.

[9] S.J. Golestani: “A Self-Clocked Fair Queueing Scheme for Broadband Applications.” *Proceedings of IEEE INFOCOM ‘94*, Toronto, June 1994, pp 636-646.

[10] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski: *Assured Forwarding PHB Group*, IETF, RFC 2597, June 1999.

[11] V. Jacobsen, K. Nichols, K. Poduri: *An Expedited Forwarding PHB*, IETF, RFC 2598, June 1999.

[12] M. Katevenis, S. Sidiropoulos, C. Courcoubetis: “Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip.” *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 3, Oct. 1991, pp 1265-1279.

[13] C. Liu, J. Layland: “Scheduling algorithms for multiprogramming in a hard real-time environment.” *Journal of the ACM*, Vol. 20, No. 1, Jan. 1973, pp. 46-61.

[14] M. Loukola, J. Ruutu, K. Klikki: *Dynamic RT/ NRT PHB Group*. Internet Draft <draft-loukola-dynamic-00.txt>, work in progress, Nov. 1998.

[15] T. Nandagopal, N. Venkitaraman, R. Sivakumar, V. Bharghavan: “Delay Differentiation and Adaptation in Core Stateless Networks.” *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, March 2000.

[16] K. Nichols, S. Blake, F. Baker, D. Black: *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, IETF, RFC 2474, Dec. 1998.

[17] S. S. Panwar, D. Towsley, J. K. Wolf: “Optimal Scheduling Policies of a Class of Queues with Customer Deadlines to the Beginning of Service.” *Journal of the ACM*, Vol. 35, No. 4, Oct. 1988, pp. 832-844.

[18] J. Ruutu, K. Kilkki: “Simple Integrated Media Access - a Comprehensive Service for Future Internet.” *Proceedings of the IFIP Conference on Performance of Information and Communications Systems (PICS ’98)*, Lund, Sweden, May 1998.

[19] V. Sivaraman, F. M. Chiussi: “Statistical analysis of delay bound violations at an earliest deadline first (EDF) scheduler.” *Performance Evaluation*, Vol. 36-37, Aug. 1999.

[20] Y. Xie, T. Yang: “Cell discarding policies supporting multiple delay and loss requirements in ATM networks.” *Proceedings of the IEEE Global Telecommunications Conference (Globecom ’97)*, Phoenix, Arizona, Nov. 1997.

[21] Z.-L. Zhang, D. Towsley, J. Kurose: “Statistical Analysis of the Generalized Processor Sharing Scheduling Discipline.” *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 6, Aug. 1995, pp 1071-1080.