Transient Performance Analysis of the Selective Drop Buffer Acceptance Scheme with Responsive Traffic

K. Spaey, C. Blondia
University of Antwerp, Department of Mathematics and Computer Science
Performance Analysis of Telecommunication Systems Research Group
Universiteitsplein 1, B-2610 Wilrijk, Belgium
{spaey,blondia}@uia.ua.ac.be

***Abstract*- This paper considers the Selective Drop (SD) algorithm, a packet aware buffer acceptance algorithm used with the Unspecified Bit Rate ATM service category. The influence of the parameters of the SD algorithm on the transient performance results (efficiency and fairness) will be studied when starting from an unfair start situation. This will be done using an analytical model where two responsive sources send their traffic in packets consisting of cells via a buffer on which the SD algorithm is implemented. Three different scheduling algorithms will be used in combination with the SD algorithm. Based on an extensive set of numerical examples, observations will be made. From these results, one of the most important conclusions is that a large improvement of the fairness results is seen when SD is implemented, irrespective of the exact setting of its parameters. On the efficiency results, these parameters have however a more important effect. So it is recommended to implement SD to increase the fairness, but with a parameter setting focussed on the efficiency results.**

I. Introduction

The importance of packet aware buffer acceptance schemes such as Partial Packet Discard (PPD) and Early Packet Discard (EPD) [1] to improve the efficiency of packet-based protocols like TCP over the Unspecified Bit Rate ATM service category, has been widely recognized by the implementation of these schemes in most commercial ATM switches. But besides efficiency also fairness between the throughput of the different connections is important. Because EPD does not take the current rate or buffer utilization of the different connections into account when discarding packets, it cannot guarantee fairness. Therefore buffer acceptance schemes, such as Selective Drop (SD), Fair Buffer Allocation (FBA) and EPD with per-VC queueing, ([2], [3], [4], [5]), which preferentially discard packets of connections that take more than their Fair Share (FS) of the buffer space, were developed.

This paper will concentrate on the Selective Drop algorithm and analyze its transient performance when traffic is generated by sources which respond to the presence or absence of losses (like TCP sources). For this goal we developed in [6] an analytical model where two responsive sources send traffic in fixed-sized packets of cells via a buffer on which the SD buffer acceptance algorithm is implemented. Transient efficiency and fairness results are obtained from the model under an unfair start condition, which corresponds to a situation where one source alone has been sending traffic for some time, and suddenly a second source starts also sending traffic, leading to a bottleneck.

Where performance oriented studies typically rely on the assumption that the stochastic process modeling the phenomenon of interest is already in steady state, transient performance analysis is important when the life cycle of the phenomenon under study is not large enough. Such a transient analysis is necessary when a stochastic process does not reach steady state, or when its behavior before reaching steady state is important. So when observing the reaction upon an unfair start situation of a buffer acceptance scheme which aims at fairness, a transient approach is required.

The main goal of this paper is to study the influence of the parameters of the SD algorithm on the efficiency and fairness results. The paper is organized as follows: a short description of the Selective Drop algorithm is given in Section II. Section III discusses the model used and how the transient efficiency and fairness results are calculated. In Section IV, observations made on an extensive set of numerical results are given and illustrated by examples. Finally, Section V draws the conclusions.

II. The Selective Drop buffer acceptance algorithm

Selective Drop ([2], [4]) is a packet aware buffer acceptance algorithm that is based on the principle that a connection that gets more than its fair share of the buffer space will also get more than its fair share of the bandwidth. A flowchart of the algorithm is shown in Fig. 1. The following notation is used: Q denotes the buffer occupancy, $Q_{\max}$ the capacity of the buffer,

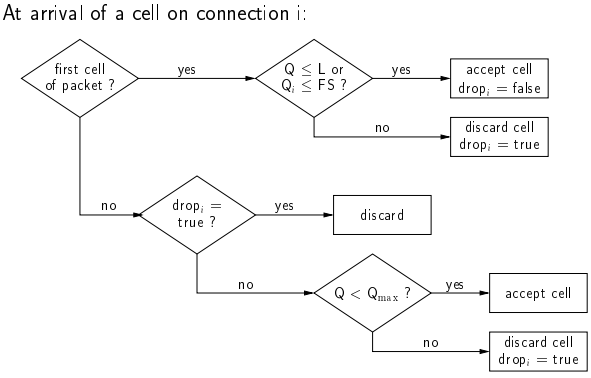

Fig. 1. Flowchart of the SD algorithm. The following notation is used: Q: buffer occupancy; $Q_{\max}$: capacity of the buffer; Q_i: number of cells of source i in the buffer; L: a fixed threshold; FS: Fair Share. FS is calculated as $(Q/N) \times K$, where N is the number of active connections and K is a fixed parameter of the SD algorithm.

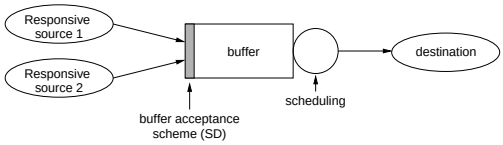

Fig. 2. System configuration

Q_i the number of cells of source i in the buffer, N the number of active connections (i.e., the number of connections that have at least one cell in the buffer), L a fixed threshold and K a parameter of the SD algorithm. As can be seen from Figure 1, a packet (i.e., all its cells) of connection i is allowed to enter the buffer if $Q \leq L$ or if $Q_i \leq$ FS, where FS denotes the Fair Share, which is defined as

$$\text{FS} = \frac{Q}{N} \times K. \tag{1}$$

The packet awareness of the algorithm appears from the fact that it tries to discard and accept complete packets (i.e., all their cells) as does the EPD scheme [1]. Because of that, the SD algorithm is sometimes also called 'EPD with per-VC accounting' [4]. Originally, the SD acceptance algorithm was defined for implementation with a global buffer on which FIFO scheduling is applied, but we will consider it also in combination with Round Robin (RR) and Probabilistic Longest Queue First (PLQF) scheduling. The combination with RR scheduling is known as 'EPD with per-VC queueing' [4].

III. Model description and transient performance measures

The performance of the SD buffer acceptance scheme will be observed using the configuration of Fig. 2.

A. Source behavior

It is assumed that the traffic in the system is generated in packets of D back-to-back cells, where D is fixed, by two independent but identical responsive sources. A responsive source responds to the presence/absence of losses of its traffic by decreasing/increasing the amount of packets it sends in a certain time. All traffic is sent to the same destination via the output port of a network element. The links in the scenario have all the same capacity, which makes this output port a bottleneck at which buffering is needed. The time needed to place D cells onto the links is considered as time unit of the system, and is called a 'slot'. On the input links, a slot thus equals the time to place a packet onto the links, while on the output link the D cells which are put onto the link in a slot can belong to both connections, depending on the output of the scheduling algorithm. Because it is known how many cells of each connection leave the buffer in a slot (see Section III-C), the time unit of the system could be chosen as D cell times. The sources are persistent sources that have always traffic to send, but the amount of packets they can send in a time of x slots (where x is a parameter of the source model) is limited by their window size. The window sizes of the sources are updated every x slots, based on the number of packets a source has lost at the buffer in the previous x slots. The following rules are used for the window updates:

- if a source has not lost any packets during the previous x slots, then its window size is increased by one packet, except if it has already reached its maximum window size of x packets,
- if a source has lost one packet during the previous x slots, then its window size is approximately halved, by setting it to the smallest integer not smaller than half its current window size,
- if a source has lost two or more packets during the previous x slots, then its window size is reduced to one packet.

Further, it is assumed that a source with a window size of r packets ($1 \leq r \leq x$) will send these packets during the first r slots of an interval of x slots.

Remark that due to the assumption that the two sources are identical, x is the same for both sources. A possible extension of the model (not considered in this paper) could be to take the window update interval x different for both sources.

B. Buffer acceptance

When a packet arrives at the buffer, the decision about if it is allowed to enter the buffer or not is made based upon the SD buffer acceptance algorithm. Because of the assumption in the model that the sources send the D cells of a packet back-to-back, packet boundaries correspond to slot boundaries. Since the acceptance rule ($Q \leq L$ or $Q_i \leq$ FS) of the SD algorithm is only tested for the first cell of a packet (see Fig. 1), a decision about the acceptance or discarding of the complete packet can be made in the model at slot boundaries. If packets from both sources arrive at the same time and they both pass the acceptance rules, but there is only place in the buffer for one packet, then it will be assumed that each packet has equal probability of being the one that will be dropped.

C. Scheduling

Three scheduling algorithms will be considered: FIFO, Round Robin (RR) and Probabilistic Longest Queue First (PLQF). In a FIFO system, if the D cells of a packet arrive back-to-back at the buffer, D cells of one connection (when *upon arrival* of this packet no packet of the other connection arrived), or $D/2$ cells of each connection (when a packet of both connections arrived at the same time) leave the buffer in a slot. In a RR system on the other hand, when at *departure instants* no cells of the other connections are present, D cells of one connection will leave the buffer in a slot. Otherwise, $D/2$ cells of each connection leave the buffer in a slot. The system will also be considered with a PLQF scheduling discipline, which selects for service a customer from a connection with a probability proportional to the contribution of this connection to the total queue length. Where the aim of RR scheduling is to let an equal amount of cells of each connection leave the buffer per scheduling cycle, PLQF scheduling strives to an equal amount of cells of each connection in the

buffer. Corresponding to the FIFO and RR system, also in the PLQF system we let $D/2$ cells of each connection or D cells of one connection leave the system in a slot, and this with the following probabilities:

- $D/2$ cells of each connection, with probability S/Q,
- D cells of connection 1, with probability $(Q_1 - S/2)/Q$,
- D cells of connection 2, with probability $(Q_2 - S/2)/Q$,

where S is the number of cells in the buffer belonging to packets that have been accepted at the same time as other packets.

D. System evolution

For RR and PLQF scheduling, the evolution over time of the systems corresponding to the source behavior, buffer acceptance and scheduling rules defined before, is described by a multi-dimensional discrete-time Markov chain. More formal details about this can be found in [6] and [7]. From the state of the system at time k, the random variables $O_i(k)$, used in Section III-E, are obtained, where $O_i(k)$ is the number of cells of connection i that leave the buffer during slot k.

For FIFO, which is one of the simplest scheduling schemes to implement, it is necessary to keep track of the order in which the cells of the different connections have entered the buffer. In an analytical model this is difficult to incorporate, since even for two sources this will make the number of states in the model very large, leading to an unattractive model. Because of that, we did not describe the system evolution of the FIFO system analytically, but results were obtained by simulation (after performing 75000 runs).

E. Transient performance measures

Both the efficiency and fairness performance of the system in transient state are of interest and can be obtained from the effective throughput $T_i(k)$ of the connections after k slots. Since the effective throughput of a connection is defined as the average number of packets of that connection that have arrived at the destination, divided by the time needed to deliver these packets, $T_i(k)$ is calculated as

$$T_i(k) = \frac{1}{Dk} \sum_{j=0}^{k-1} E\big[O_i(j)\big]. \tag{2}$$

The efficiency after k slots is defined as the sum of the effective throughput of all connections after k slots, divided by the maximum possible effective throughput after k slots (which is one packet per slot time), resulting in

$$\text{efficiency}(k) = T_1(k) + T_2(k). \tag{3}$$

To decide about the fairness performance of the system, the fairness index defined by the ATM-Forum in [8] will be used. Since the equal division of the total effective throughput among both connections will be considered as the perfectly fair situation, the fairness index after k slots, denoted by $F(k)$, is given by

$$F(k) = \frac{\big(T_1(k) + T_2(k)\big)^2}{2\big(T_1(k)\big)^2 + 2\big(T_2(k)\big)^2}. \tag{4}$$

TABLE I

PARAMETER SETTINGS FOR THE SYSTEMS CONSIDERED IN SECTION IV.

x (slots)	$Q_{\max}$ (cells)	$Q_{\max}/x$	maxtime (slots)
6	$5 \times D$	0.83	900
	$7 \times D$	1.17	
	$12 \times D$	2.00	
10	$5 \times D$	0.50	1500
	$8 \times D$	0.80	
	$12 \times D$	1.20	
	$20 \times D$	2.00	
13	$7 \times D$	0.54	1950
	$10 \times D$	0.77	
	$16 \times D$	1.23	
	$26 \times D$	2.00	

Remark that for two sources, $F(k)$ ranges between one half (minimum fairness) and one (maximum fairness).

IV. RESULTS AND DISCUSSION

Using the model described in Section III, we illustrated in previous work [6] with examples that, due to the responsiveness of the sources, it is not necessarily true anymore that being more conservative in accepting packets implies a lower efficiency, as would be the case when non-responsive sources would be used. Also the fairness of some examples was addressed, showing that there is not necessarily a trade-off between efficiency and fairness.

In this paper, the influence of the parameters of the SD algorithm on the efficiency and fairness results will be studied. It is assumed that the buffer is empty at time 0, while the window setting is the most unfair situation possible, i.e., the window of connection one is at its maximum, that of connection two at its minimum. This start condition can be seen as a situation where only one source has been sending traffic for some time, and because there was no bottleneck, all traffic of this source could pass through the system without building up a queue and without losses. The window size of this source could thereby grow until its maximum. At time 0, the second source starts also sending traffic, starting with a window size of one packet. In real systems, unfair (start)situations will constantly be created when connections appear and disappear. Because of the unfair start situation, all fairness curves have a typical shape. In the beginning they go very fast down, because then an unfair amount of packets of each connection is offered to the system; since the buffer is empty then, all packets are accepted until Q exceeds L, so also the output of the system is unfair in the beginning. Afterwards, the fairness increases again in a rather steep and fluctuating way, and finally it slowly grows towards one. When discussing fairness results further on, this last part of a fairness curve will be called the 'horizontal' part, the other part the 'steep' part.

A. Influence of the SD parameters: the threshold L

The scenarios with parameters as shown in Table I and the SD parameter K taken equal to 1 are considered. The threshold L is varied between $L = 1 \times D$ and $L = Q_{\max} - D$. Remark that the maximal setting of L corresponds to the case where the SD algorithm is not implemented, since at the moments that packets arrive, there is always place in the buffer for at least one packet, because D cells have just left it. So when L is set at D cells before $Q_{\max}$, it is always true that $Q \leq L$ when packets arrive, and the test $Q_i \leq$ FS is never performed. The following observations are made based on the results:

- For RR and PLQF scheduling, the efficiency generally increases when L increases. This seems natural because increasing L implies that more packets will be accepted, but as has been mentioned before, this is not always true due to the responsiveness of the sources. The exceptions to this general rule are:
  - There are always settings of L through which higher efficiency values are obtained than when L is set to its maximal value $Q_{\max} - D$ (i.e., SD is not implemented). With RR scheduling, there are more of these settings than with PLQF scheduling. Sometimes even perfect efficiency values (i.e., constantly equal to 1) are obtained with RR. With the implementation of the SD algorithm, whose main intention is to increase the fairness, there are thus settings of L that allow to obtain also a higher efficiency than when SD is not implemented.
  - With RR scheduling, in case that the efficiency results obtained are very high, it is possible that a larger L leads to a lower efficiency. Probably because these results are so close to optimal, a change of L becomes less significant.
  - With PLQF scheduling, for efficiency results which are among the highest obtained with a particular scenario, sometimes a larger L gives lower efficiency results.
  - A few examples are found with RR scheduling where the efficiency is drastically lower than what would be expected when looking at the results obtained with neighboring examples (i.e., examples where the difference in the setting of L is only D cells). In these examples the windows of both sources synchronize after a while, but in such a way that the buffer becomes often empty, which pulls the efficiency down. None of such examples occur with PLQF scheduling, because of the probabilistic character of such systems.
- With FIFO scheduling, the statement that the efficiency increases when L increases is true when x is small ($x = 6$), and for very small values of L for the other x's. In the other cases, no real relation can be found between a change of L and the corresponding change of the efficiency, but in general large L values (a few packet sizes before the end of the buffer) give better efficiency results than small L values. As with RR and PLQF scheduling, also with FIFO scheduling there are always settings of L with which higher efficiency values are obtained than when SD is not implemented, but sometimes these results are not the whole time above these obtained when no SD is implemented, but only in the long run. As with RR scheduling, also with FIFO scheduling some examples are found where the windows of both sources synchronize in such a way that the buffer becomes often empty, implying decreasing efficiency results.
- A main observation that can be made about the fairness for the systems with RR and PLQF scheduling is that it is always much better when SD is implemented than when SD is not implemented, irrespective of the exact setting of L. For all settings of L such that $L < Q_{\max} - D$ (i.e., SD implemented), no specific setting of L can really be judged to give results that are the whole time better than with another L. With FIFO scheduling, in general the same observation can be made. However, a few exceptions are found now where the fairness is worse in a scenario where SD is used than when it is not used.
- In the very beginning, the fairness curves coincide for all L, since the behavior of all systems is the same as long as $Q \leq L$. Later, the curves split. The smaller L, the sooner a curve splits from the other curves, since the smaller L, the sooner the SD scheme starts to solve the initial unfairness.
- In general, the larger x, the longer the steep part lasts, when time is expressed in multiples of x. This indicates that the longer between adapting the windows, the longer it takes before the initial unfairness is more or less solved.

B. Influence of the SD parameters: the parameter K

For the scenarios of Table I with $x = 10$ and different settings of L, K will now be chosen from $K = 1$, $K = 1.2$ and $K = 1.4$. The larger K, the less severe the SD algorithm is in dropping packets. The following observations are made based on the results:

- With RR scheduling, it is true in general that when K increases, then the efficiency stays equal or increases also. The larger L, the smaller the positive effect of increasing K becomes. Some exceptions are found where the efficiency obtained with $K = 1.2$ or $K = 1.4$ is the lowest. This occurs when the efficiency results are very large or in scenarios where the windows synchronize in such a way that the buffers become often empty. On the fairness results almost no influence of K is noticed. The steep parts of the fairness curves for the different K mostly coincide, since as long as cells are present in both queues, the output of the RR scheduling algorithm is the same for the different scenarios.
- With PLQF scheduling, for small L (approximately $L < x/2$) a larger K gives a larger efficiency. The larger $Q_{\max}$ is, for the larger L values this stays true. When L is increased, the results evolve through the following situations: (i) a larger K gives still a larger efficiency in the long run, but in the transient phase the efficiency curves cross. (ii) $K = 1.2$ gives a higher efficiency than $K = 1$, but for $K = 1.4$ the efficiency is below that obtained with $K = 1$, (iii) a larger K implies a lower efficiency. For the fairness, some differences are noticed when changing K, but the different fairness curves still stay very close to each other. The largest difference is noticed in the steep parts of the curves, where the smallest K value gives

the best result.

- When FIFO scheduling is applied, as with PLQF scheduling the efficiency increases when K increases for small L. The more L grows, an evolution towards the fact that a larger K gives a lower efficiency is seen. Concerning the fairness results, curves obtained for different K values coincide in the beginning, after which they one by one branch off. In the steep part the best fairness is obtained when K equals 1. In the long run, it is difficult to judge which K value gives best results in a scenario. What is seen often in the horizontal parts of the fairness curves is a slowly oscillating behavior. Curves which show this behavior correspond often to scenarios with which perfect efficiency values are obtained.

C. Examples

Due to lack of place, only a very few of the observations made in Sections IV-A and IV-B are illustrated here with examples. For more of these illustrations, we refer to [7].

Figs. 3 and 4 show some results obtained when PLQF scheduling is used. In Fig. 3, efficiency results are shown for $x = 13$ slots and $Q_{\max} = 16 \times D$ cells. This figure illustrates that there are settings of L with which higher efficiency values are obtained than when SD is not implemented ($L = 15 \times D$). Further it can be seen from the figure that in general, the efficiency increases when L increases, although this is not always the case. For example, for $L = 11 \times D$ the efficiency is larger than for $L = 13 \times D$. When looking at the *most-likely path* (which is the path obtained by following always the branch with the highest probability when the sample path of the system evolution splits) for this last scenario, it is seen that from a certain time on (around 420 slots) approximately once every 520 slots, the window of one connection is forced down until its minimum, while that of the second connection, which at that time was not too large, is halved, such that both connections end up with a small window. Some time is needed to let these windows grow again, during which the buffer flows empty for a few slots. Fig. 4 shows fairness results when x equals 10 slots and $Q_{\max}$ is $20 \times D$. The figure illustrates clearly that the fairness obtained when SD is not implemented ($L = 19 \times D$) is worse than when it is implemented and that the fairness curves coincide in the beginning, and branch off one by one, first for the smallest L. This branching off happens sooner here than in the corresponding case with RR scheduling, since with RR a difference in fairness occurs only from the moment that there is a difference in the output for the scenarios with different L. This happens when there is a difference in which queue is empty at the particular moment. Remark that the curves shown in Fig. 4 (and also in Fig. 6) show only a fraction of the time 'maxtime' since as soon as a system has solved the initial unfairness, the fairness index converges towards one, because the behavior of the two sources considered is the same, and they are treated equally by the buffer acceptance and the scheduling algorithm. This illustrates again the importance of a transient analysis when observing the behavior

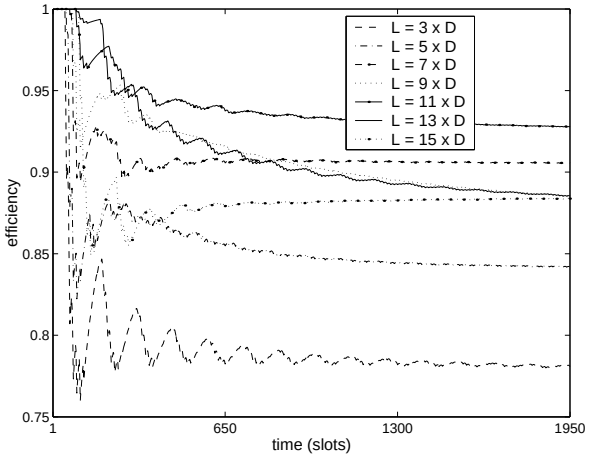

Fig. 3. Efficiency results obtained when $x = 13$, $Q_{\max} = 16 \times D$ and $K = 1$ for different settings of L (PLQF scheduling).

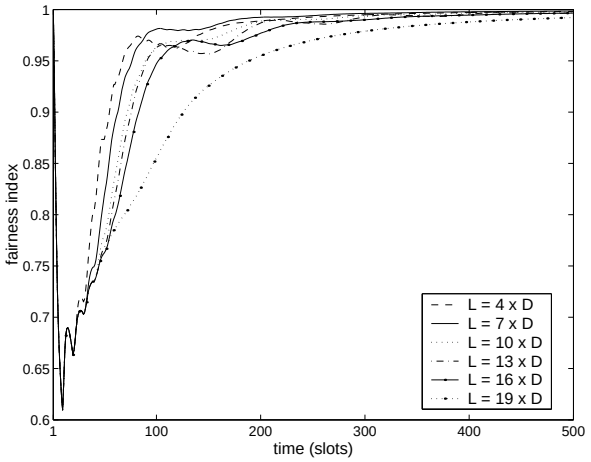

Fig. 4. Fairness results obtained when $x = 10$, $Q_{\max} = 20 \times D$ and $K = 1$ for different settings of L (PLQF scheduling).

of the SD scheme towards an unfair start situation.

Figs. 5 and 6 show results for the different values of K when FIFO scheduling is applied. In Fig. 5 efficiency values are shown for a scenario in which $x = 10$ and $Q_{\max} = 12 \times D$. As can be seen, when L equals $3 \times D$, then the efficiency increases when K increases. For $L = 7 \times D$, the efficiency is the largest (i.e., perfect) when $K = 1.2$. The efficiency obtained with $K = 1$ is larger than that obtained with $K = 1.4$, while in the transient phase curves cross. When L equals $9 \times D$, the lowest efficiency is obtained when $K = 1.4$. Fig. 6 shows fairness results for different K when L equals $9 \times D$ for the same scenario as used in Fig. 5. The three fairness curves coincide in the beginning and then branch off one by one. In the horizontal part, the curves for $K = 1$ and $K = 1.2$ show an oscillating behavior caused by the fact that the window of one connection is high while that of the other connection is low. Because of the FIFO scheduling, this implies that during a period more cells of connection one will leave the buffer,

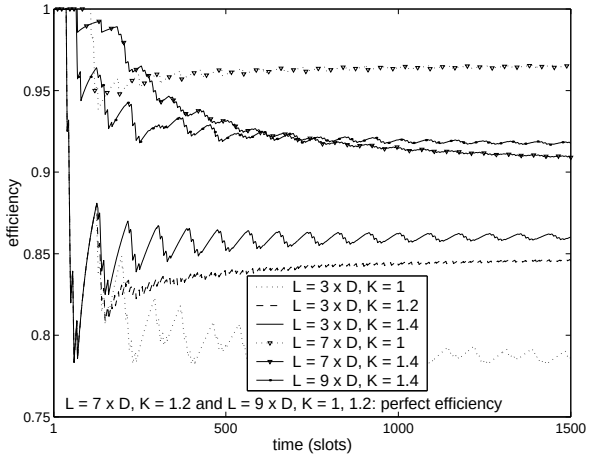

Fig. 5. Efficiency results obtained when $x = 10$, $Q_{\max} = 12 \times D$ for different settings of L and K (FIFO scheduling).

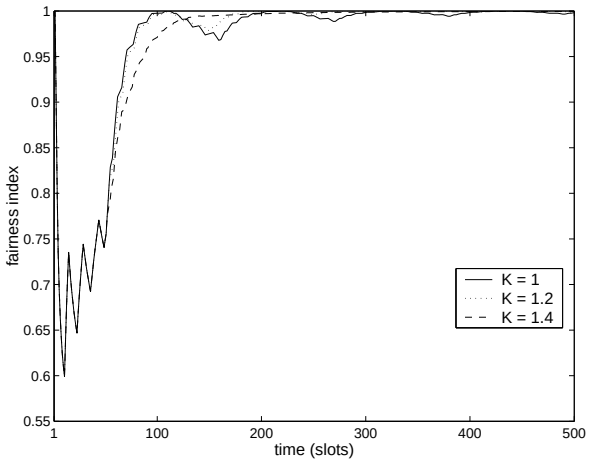

Fig. 6. Fairness results obtained when $x = 10$, $Q_{\max} = 12 \times D$ and $L = 9 \times D$ for different settings of K (FIFO scheduling).

such that the fairness goes down. During a following period, more cells of connection two leave the buffer, such that the fairness grows again, and so on. Comparing with Fig. 5 learns that perfect efficiency values are obtained when $L = 9 \times D$ and $K = 1$ or $K = 1.2$.

V. Conclusions

In this paper the influence of the parameters of the Selective Drop algorithm on the transient efficiency and fairness results was studied using an analytical model. In this model (i) traffic is sent in packets of cells towards a buffer by sources which respond to the presence or absence of losses, (ii) packets are accepted or discarded based on the SD algorithm, and (iii) cells leave the buffer in an order determined by the RR, PLQF or FIFO scheduling algorithm.

The most important conclusion of this study is that the presence of the SD algorithm has a large positive effect on the fairness results, irrespective of the exact setting of the parameters of the algorithm. On the efficiency results however, these parameters have more influence.

With RR and PLQF scheduling, the efficiency generally increases when the threshold L increases, and choosing L at a few packet sizes less than the size of the buffer results in a good setting. With RR scheduling, the chance is rather high that the efficiency values obtained are then even above these obtained when SD is not implemented (so there is no trade-off between efficiency and fairness then). With PLQF scheduling, this chance is reasonable. Remark however that with RR scheduling, sometimes the efficiency is lower than expected because of synchronization effects. When using PLQF scheduling, no lasting synchronization will occur because of the probabilistic character of the scheduling algorithm in these scenarios. Also with FIFO scheduling, synchronization can occur. With FIFO scheduling it is much harder to make a conclusion about the best setting of the threshold L, since no real relation was found between a change of L and a corresponding change of the efficiency. But choosing it a few packet sizes less than the size of the buffer as with RR and PLQF scheduling gave in most scenarios rather good results.

The parameter K of the SD algorithm has also more influence on the efficiency results than on the fairness results. Increasing K has principally a positive effect on the efficiency when L is set at a small value. When the setting of L is larger, this positive effect is still seen with RR scheduling, but with PLQF and FIFO scheduling the probability is rather high that the efficiency will be lower than when K is chosen equal to one.

As a general conclusion, it is recommended to implement SD to increase the fairness, but with a parameter setting focussed on the efficiency results. Good efficiency results are obtained when the threshold L is set a few packet sizes less than the buffer size, and when the parameter K of the SD algorithm is chosen equal to one.

References

[1] A. Romanow and S. Floyd, "Dynamics of TCP traffic over ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, May 1995.

[2] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy, and S.C. Kim, "UBR+: improving performance of TCP over ATM-UBR service," *Proceedings of ICC'97*, June 1997.

[3] J. Heinanen and K. Kilkki, "A fair buffer allocation scheme," *Computer Communications*, vol. 21, 1998.

[4] H. Li, K. Siu, H. Tzeng, C. Ikeda, and H. Suzuki, "On TCP performance in ATM networks with per-VC early packet discard mechanisms," *Computer Communications*, vol. 19, 1996.

[5] K. Spaey and C. Blondia, "Buffer acceptance schemes for the UBR and GFR service categories," *Proceedings of ATM & IP 2000*, July 2000.

[6] K. Spaey and C. Blondia, "Observations of the transient performance of the selective drop buffer acceptance algorithm with responsive traffic," *Proceedings of ATM & IP 2001*, June 2001.

[7] K. Spaey and C. Blondia, "Transient performance analysis of the selective drop buffer acceptance scheme with responsive traffic," *Available at ftp://win-www.uia.ac.be/pub/pats/reports/icccn_ev.ps*

[8] ATM Forum, *ATM forum performance testing specification*, October 1999.