Analysis of a delay based congestion avoidance algorithm

W. Dabbous

INRIA Centre de Sophia Antipolis, 2004 Route des Lucioles, BP-93, 06902 SOPHIA-ANTIPOLIS Cedex, FRANCE.

Abstract

There has been considerable interest recently on flow control mechanisms where the data flow into a network is regulated based on feedback from the network. The window mechanism developed by Jacobson, in which sources use packet losses to adjust their window sizes, resulted in dramatic reduction of congestion in the Internet and has become an Internet standard. Recent studies have shown that Jacobson's algorithm gives rise to large-amplitude oscillations of window sizes and packet delays, which make it unsuitable to support applications such as packet video. In this paper, we describe and evaluate the performance of a flow control mechanism in which sources use packet round trip delays to adjust their window sizes. The objective is to maintain packet delays at a target level close to the minimum possible delay, and yet not severely restrict throughput. Simulation and experimental results show that the mechanism reduces the oscillations of window sizes, and provides connections with low average delay and low delay jitter, thus making it suitable to support applications such as a videoconference application currently under study, provided that some reservation mechanism is used in the intermediate gateways.

1 Introduction

In a computer network, packets generated by a source node are delivered to their destination by routing them via a sequence of intermediate nodes. If the source rates are increased without constraint, queues of packets waiting to be routed build up at bottleneck nodes, leading to high delay. Eventually, the buffering capacity of these nodes is exceeded and packets are dropped, resulting in low throughput as well. Flow control mechanisms attempt to maximize throughput and minimize delays and losses by imposing constraints on the flow of data into the network.

Consider a source-destination pair whose packets are routed via intermediate nodes. Let μ_i be the service rate offered by intermediate node i to packets of this source-destination pair. For two nodes i and j, μ_i and μ_j can differ for several reasons: they may be connected to links with different bandwidths; the total traffic through the nodes may differ because they support different source-destination pairs; their hardware may differ; etc. If the network is in steady-state, the ideal flow control policy is to limit the source rate to the *bottleneck rate* $\mu = \min_i(\mu_i)$ [9], or equivalently the interpacket gap to $1/\mu$; a higher rate would result in packet delay or loss, and a lower rate in underutilization.

In reality, however, the bottleneck rate changes with time because connections (i.e., source-destination pairs) are being set up and terminated, and because their sources do not maintain constant data rates. Recently, flow control mechanisms have been proposed where the source rate is dynamically regulated based on feedback from the intermediate nodes [4, 11, 14, 17, 19]. These so-called end-to-end mechanisms attempt to adapt the source rate to the bottleneck rate in minimum time and with minimum packet delay or loss. Other mechanisms, referred to as gateway flow control mechanisms, determine which packets are discarded when the buffering capacity of a node is exceeded, and the order in which buffered packets are sent [16]. Gateway control mechanisms determine the way in which packets from different sources interact with each other, which in turn affects the behavior of end-to-end mechanisms. They are still the subject of active research (e.g. [15, 22]).

The rest of the paper is organized as follows. In the next section, we describe in more detail the issues concerning the congestion control problem, focusing on the analysis of the standard slow start algorithm. In section 3, we describe a delay based flow control mechanism in detail. In section 4, we study the performance of the delay based algorithm via simulation and compare with slow start. Our results show that the mechanism provides low average delay and low delay variance in a homogeneous environment, i.e. a network in which all connections use the same flow control mechanism. In section 5, we describe modifications which improve the ability of the mechanism to respond to changes in network conditions, e.g. route changes. In section 6, further simulation results indicate that the mechanism would be adequate to support video applications in heterogeneous networks with so-called rate allocating gateways, such as Fair Queueing gateways [6]. Section 7 concludes the paper.

2 Congestion avoidance and control

In this section, we focus on end-to-end mechanisms. One of the most important characteristics of such a mechanism is its feedback: What information does the source obtain about the state of the intermediate nodes. In the binary feedback scheme used in DECNET [18], each data packet received by the destination has a bit indicating whether or not the packet encountered an intermediate node with average queue size greater than 1; this information is sent back to the source in acknowledgement packets. In the Internet TCP, the feedback consists of the arrival times (or lack of arrival) of acknowledgement packets [11]. Other schemes rely on feedback information that includes average link utilization [19], interarrival times between successive acknowledgement packets [14], packet round trip delays [17], [12] etc.

The feedback information is used to regulate the flow of data from the sources into the network. In *rate-based* mechanisms, the source rate is regulated by adjusting interpacket gaps. Examples are found in the protocols NETBLT [4], VMTP [2], and XTP [3]. However, most protocols currently in operation [18, 11] use a *window-based* scheme, rather than the rate-based scheme. Here, a limit referred to as the *window size* is placed on the number of packets that can be outstanding at the source, but no constraint is placed on the rate at which these packets can be sent. For example, in the binary feedback scheme, the source decreases the window size by a multiplicative factor if the more than a fraction of acknowledgement packets have their feedback congestion bit set to 1; otherwise, it increases the window size linearly. Slow Start is a congestion avoidance and control scheme defined by Jacobson [11]. In this scheme, the source increases the window when an acknowledgement is received, and decreases the window when a packet loss is detected. Specifically, the window adjustment mechanism has two phases, the slow-start or congestion recovery phase, and the linear increase or congestion avoidance phase.

At connection set-up, the window is set to one[1] and the slow-start phase begins. The window is increased by 1 whenever an acknowledgement is received, until it reaches a threshold value, at which point the algorithm switches into the congestion avoidance phase. There, the window is increased by $1/W$ whenever an acknowledgement is received, where W denotes the current window size. When a packet loss is detected[2], the window is set to 1 and the slow start phase resumes; the idea being that a packet loss means that the queues at the intermediate nodes are overflowing.

In Jacobson's scheme, queue sizes at the intermediate nodes are kept relatively large, in case of FCFS (First Come First Serve) service discipline. It is also intuitively clear, and it has been verified in numerous simulation (e.g. [20]) and experimental studies (e.g. [11]), that the evolutions of the window size as a function of time follow a cyclic pattern. In many cases, the amplitudes of the window oscillations turn out to be quite large.

The above observations imply that queueing delays, and hence packet delays, follow a cyclic behavior with large maximum value and large variance. This makes Jacobson's algorithm unsuitable to support applications such as video image transmission (e.g. video conference) which require relatively low average delay and low delay jitter. Furthermore, we note that the periodic reduction of the window size to one following a packet loss will result in periodic blackouts of the video signal.

The goal of our study is to choose a congestion control algorithm suitable to be used by the TPX protocol currently under design within the OSI95 ESPRIT project[7]. Therefore we will focus on transport level procedures and consider only end-systems mechanisms which we will compare in order to propose an adequate algorithm. Gateway congestion control algorithms should be studied in the framework of a general architectural design for high speed network layer extension. In most of the rest of this paper we will consider that gateways implement FCFS discipline and make the comparisons of end-system algorithms without any assumption on the existence of a special congestion control algorithm in intermediate gateways. We would like to get the best performance (efficiency and fair resource sharing) by the use of end-systems mechanisms.

3 The delay algorithm

The delay based mechanism is inspired from early work of [12] and from [10]. In this mechanism, sources use packet round trip delays to adjust their window sizes. The objective of the algorithm which we refer to as the Delay Algorithm, or DA is to maintain packet delays at a target level, which we take to be the smallest possible delay (i.e. the delay corresponding to a "no queueing case"), and yet not severly restrict throughput.

The Delay Algorithm was first proposed in [10] and [5]. It has two phases, like Jacobson's algorithm. At connection set-up and after a packet loss is detected, the window is set to 1 and the "slow-start" phase begins. The window is increased by 1 whenever an acknowledgement is received, until it reaches a threshold value, at which point the congestion avoidance phase takes over. This threshold value is determined by a condition on the *normalized delay gradient NDG*

[1] We assume that window sizes are expressed in terms of maximal packet sizes.

[2] A packet loss is detected either by expiration of a retransmission timer or the reception of duplicate acknowledgement packets.

as defined in [12]:

$$NDG = \frac{\Delta D/\Delta W}{D/W}$$

where ΔD is the increase in the round trip delay corresponding to an increase of ΔW of the window. If NDG is smaller than 0.5 the algorithm switches from the exponential phase to congestion avoidance phase[3]. The 0.5 value corresponds to "medium" gain in the delay increase.

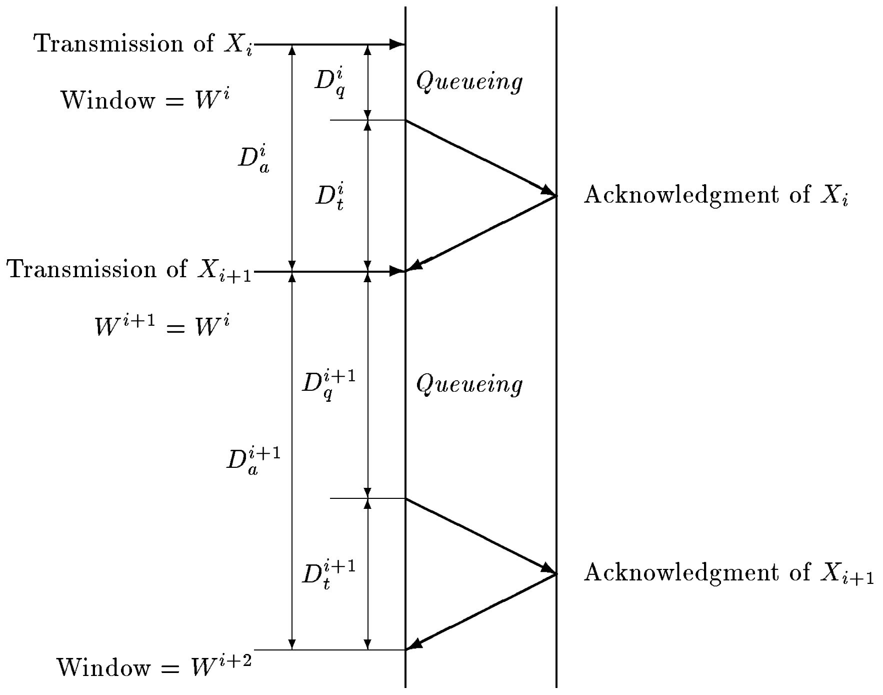

Figure 1: Correspondence between window size and round trip delays

During the congestion avoidance phase, the principle behind the DA is illustrated on Figure 1. We divide time into epochs of length equal to one round trip delay. We refer to the first packet sent in an epoch as a 'head of window'. Let X_i denote the head of window for epoch i. The source measures the round trip delays of the X_i's. Let D_a^i denote the round trip delay of packet X_i. D_a^i can be decomposed into a fixed delay D_t^i including propagation and transmission delays, and a variable queueing delay D_q^i. Let us assume for now that the fixed part of the delay does not vary between two successive epochs, i.e.,

$$D_t^i = D_t^{i+1} = D_t$$

This implies, for example, that no route or topology change occurs between two epochs[4]. D_q^{i+1} is the queueing delay caused by the $W^i - 1$ packets sent between X_i and X_{i+1}. Thus, the delay D_a^{i+1} represent the time necessary to transmit W^i packets through the network. The same throughput could have been achieved with a window:

$$W = \frac{W^i.D_t}{D_a^{i+1}}$$

[3]The numerical values for the control parameters were chosen intuitively. Their value should be adapted according to the status of the network.

[4]A modified version of the Delay Algorithm which handles such changes is described in section 5.

This window corresponds to a "zero queueing" condition.

However, the source does not have *a priori* knowledge of D_t, the round trip delay in the absence of queueing delays. It is estimated with the the smallest round trip delay seen so far, i.e. $\min\{D_a^{i+1}\}$. Thus, at epoch $i+1$, the source first estimate D_t as

$$D_t^{i+1} = \min\{D_t^i, D_a^{i+1}\}$$

then compute the new window size:

$$W^{i+2} = \begin{cases} \frac{W^i.D_t^{i+1}}{D_a^{i+1}} & \text{if } D_a^{i+1} > K.D_t^i \\ W^i & \text{if } D_t^i < D_a^{i+1} < K.D_t^i \\ W^i + 1 & \text{otherwise} \end{cases}$$

That is, we increase the window by 1 when we suppose that no queueing¡ at all occurred. Otherwise, if the delay has "increased substantially" ($D_a^{i+1} > K.D_t^i$, $1.125 \leq K \leq 1.25$) the window size is decreased by a multiplicative factor proportional to the increase in the acknowledgement delay. Note that the hypothesis of a constant transmission delay is somewhat antagonistic with the philosophy of connectionless networks, where packets belonging to the same transport connection may follow different routes. We use a factor K in order to filter small fluctuations in the round trip delay. In section 5.1, we propose a variant of the algorithm that is less sensitive to delay changes over a datagram network.

Indeed, the window W that we have computed here is a maximum value. If for some reason the other side decides to restrain the flow of credits to a value of W_c, e.g. when processing the data takes longer than transmitting them, the transmitting maximum window size will be the minimum of both W and W_c.

The idea of using the delay gradient as a feedback signal was also proposed by Zheng and Crowcroft who described in [21] an algorithm similar to the DA: a delay threshold $D_i = D_p + \alpha Q_{max} M/\mu$ is defined, where D_p is the round trip propagation delay i.e. without queueing), Q_{max} is the maximum queue length, M the packet size and μ the bottleneck capacity. D_i can be rewritten as $(1-\alpha)D_{min} + \alpha D_{max}$ where $D_{min} = D_p$ and $D_{max} = D_p + Q_{max}M/\mu$ are respectively the minimum and maximum round trip delays. The α parameter defines the operating point i.e. the ratio of the target queue size to the maximum queue size at the bottleneck. The window adjustment is based on this threshold value: during normal resource probing phase, i.e. when the window size is increased by one every round trip delay, if $D > D_i$ the congestion window is set to 7/8 of the current size. If a packet loss is detected the slow start algorithm is used. Other delay-based mechanisms have been described in [12] and [17].

In section 5 we present a version of the Delay Algorithm allowing to easily adapt to intrinsic network configuration changes and to have improved fairness between competing sources.

4 Simulation Results

In this section, we evaluate the performance of the delay algorithm using simulation. We used a modified version of the REAL simulator [13].

4.1 The simple case

We present the simulation results for a simple network topology shown in Figure 2: a "stream" source sends continuously packets of the same size (1000 bytes) to a destination. The returned acknowledgments packets are 40 bytes long. Each node has a maximum buffer capacity of 15 packets unless otherwise mentioned. The link between the source and the gateway has a 8 Mbps with a propagation delay of 5 ms. The link between the gateway and the destination have a capacity of 800 Kbps and a propagation delay of 100 ms (the bottleneck link). The total path length is then 221.44 ms and a window of 23 is sufficient to fill the pipe. The TCP connection has a maximum window size of 64 packets ($> 23 + 15$) in order to allow for a single connection to saturate the gateway buffering capacity. The simulations last for a period of $100s$. Recent measures on the Internet [1] have shown that most FTP connections last for only a short period of time[5]. However, recall that we intend to use the Delay Algorithm to support video packet transmission. It is reasonable to expect video conference applications to be active for at least a few minutes.

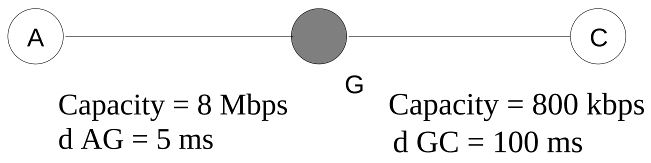

Figure 2: Network configuration with a single user

The results of the simulation are shown in Figure 3. The Jacobson-Karels[6] algorithm (JK) opens the connections with slow start and continue to increase the window until a packet loss occurs. It then enters consecutive cycles of slow start and linear increase phases, the window size oscillating between half the maximum size of 18 and the maximum size of 37 during the linear phase.

The Delay Algorithm follow the same policy as slow start for the connection start up. However, the exponential increase is stopped and the linear phase is entered at $W = 13$ (corresponding to $NDG < 0.5$) i.e. before a packet loss occurs.

The window will then stablize around 22 or 23 which corresponds to the path length with "zero queueing". The simulation also show that a DA source operates the bottleneck at maximal capacity (800 Kbps) i.e. the source has a throughput of 100 packets per second, while a JK source transmits roughly 90 packets per second. We observe that the delays for the DA source stabilize around 260 ms, while the delays for the JK source vary between 221 ms and 370 ms. Refer to Table 1.

We next compare the JK and DA algorithms in the case of two users sharing the same bottleneck link and we show how the delay algorithm eliminates the "traffic phase effects" observed in [8].

[5]For example, 90% of the TCP connections exchange less than 10Kbytes of data.

[6]The sources implementing Jacobson and Karels algorithms within the REAL simulator are labelled as "JK" sources.

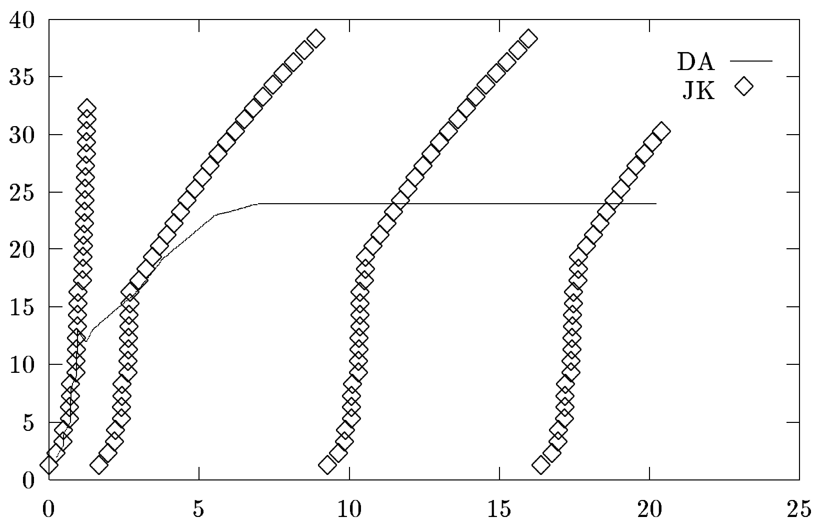

Figure 3: Window size vs time (in s) for both JK and Delay algorithms

	Minimum	Average	Maximum
DA source	260 ms	260 ms	260 ms
JK source	221 ms	287 ms	370 ms

Table 1: Round trip delays for network topology depicted in Figure 2

4.2 Competition between two sources

It has been noticed in [8] that the JK algorithm may lead to unfair resource allocation between two competing users for some network configurations with FCFS drop tail gateways[7]. This discriminatory behavior of the network is a function of two factors: the synchronous transmission of new packets driven by the service of previous packets at the bottleneck gateway, and the relative value of the round trip time of any two sources sharing the bottleneck gateway. In fact, it was shown by simulations ([8]) that a small change in the round trip time of a particular connection may cause a significant reduction in its effective throughput.

This effect (called the "phase effect") is due to an inherent characteristic of the JK algorithm: users always increase their demand (window size) until a packet loss is detected. In addition, the packet arrivals at the bottleneck gateway are related to packet departures, therefore packet arrivals for a given connection may always precede the arrivals from another connection. When the gateway buffers are full, any increase in the window size will lead a packet loss. Due to the "phase" effect, the gateway may systematically drop a packet of the same connection at the end of each congestion avoidance phase (the last packet in the gateway queue) resulting in the biased network behavior shown in simulations.

The Delay algorithm does not present such "phase effect" related unfairness. The competing

[7]When their input queue is full, drop tail gateways drop all incoming packets.

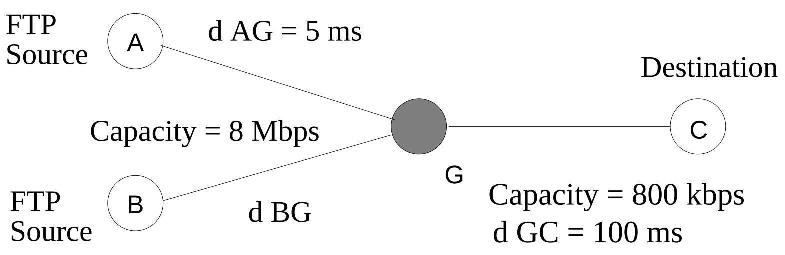

Figure 4: Network configuration with two "stream" sources

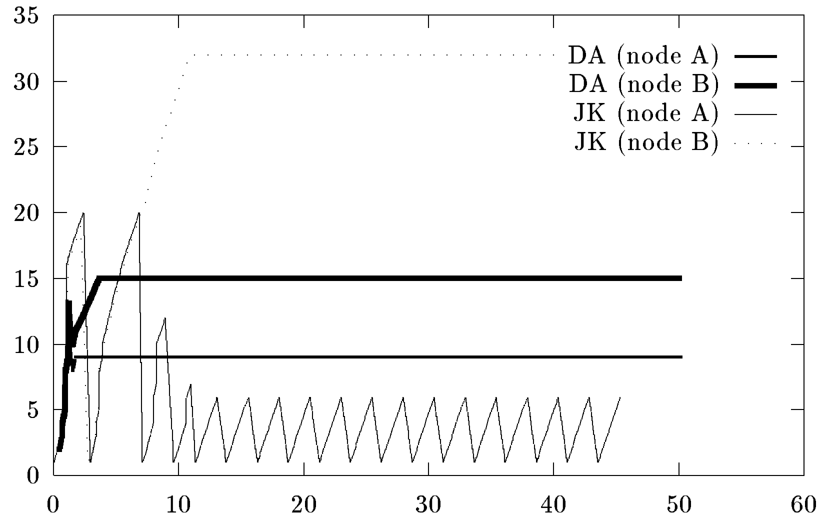

Figure 5: Window size for nodes A and B for both JK and DA algorithms, $(d_1/d_2 = 0.962)$

sources adjust their window according to feedback information about round trip times and not after a packet loss. This will prevent any systematic drop of packets of a given connection at the bottleneck gateway. This feature of the DA algorithm was verified by simulations, where two "stream" sources share a bottleneck gateway. The network topology is shown in Figure 4. The links between the sources A and B and the gateway have respectively 5 ms and d_{BG} propagation delay. For the same packet and acknowledgment sizes (1000 and 40 bytes), the total path length without queueing is $d_1 = 221.44\,\text{ms}$ for $A \rightarrow C \rightarrow A$ and $d_2 = (211.44\,\text{ms} + d_{BG})$ for $B \rightarrow C \rightarrow B$. Each source has a maximum window of 32 packets.

The simulation results in Figure 5 correspond to $d_{BG} = 9.8\,\text{ms}$ i.e. $d_1/d_2 = 0.962$. Even with such a small difference in the round trip propagation delays for both connections, the simulation shows that when the two sources are implementing the JK algorithm, node B gets a significant part of the available bandwidth. Even if we increase the buffer space at the bottleneck gateway this biased behavior persists confirming the unfair allocation of resources according to the relative values of the round trip delays. Small variations of the round trip delay may cause node's B throughput to fall to about 10 % of the total throughput.

The results in Figure 6 corresponds to a propagation delay between the source B and the gateway $d_{BG} = 3.5\,ms$, i.e. $d_1/d_2 = 1.013$. In fact, the unfairness in resources allocation between the

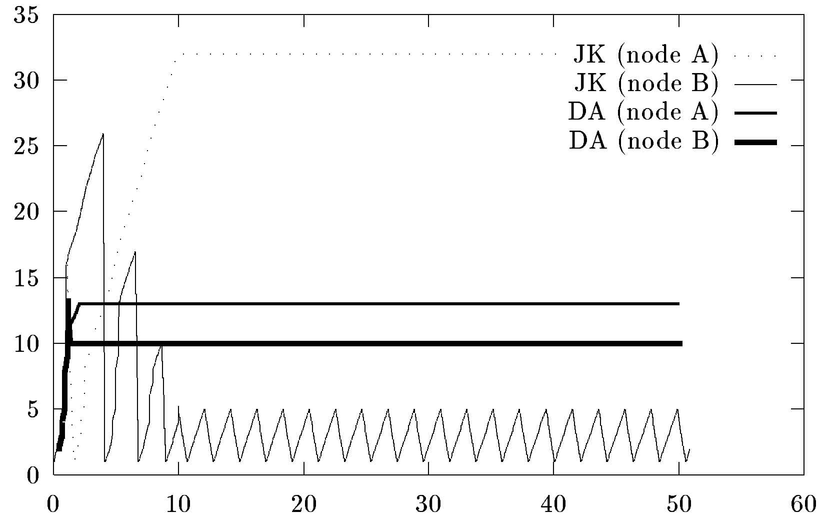

Figure 6: Window size for nodes A and B for both JK and DA algorithms, ($d_1/d_2 = 1.013$)

two sources with the JK algorithm is due to the systematic drop of the packets belonging to one of the two connections due to the phase effect. The DA shows improved fairness because there is no systematic packet drop at the bottleneck gateway. Extensive simulations confirmed these results: the DA is not sensitive to small variations of the round trip delay even with drop tail FCFS gateways. However, we might have stable “slightly unfair” sharing of the resources due to the very conservative increase/decrease policy. Refer to Figures 5 and 6: DA source A has a window size of 9 and 13, while DA source B has a window size of 15 and 10 respectively. In section 5, we describe how to tune the Delay Algorithm in order to eliminate such discriminatory behavior of the network.

4.3 Adapting to changes in network resources

The delay based feedback adjustment of the window size allows to pick the released network resources up rapidly. A load decrease due to the termination of a network connection translates to smaller delays and will be therefore noticed by active sources within a round trip delay period. In Table 2 we show the effective throughput for nodes A and B in a network configuration close to that depicted in Figure 4, with a propagation delay $d_{BG} = 5\,ms$, but with an additional “background” source (Bg) which transmits to the destination C via G at a constant rate (50 packets/s) for 40 seconds and then stops for 40 seconds. Two simulations for 120 seconds were run with both sources having the same type (JK or DA) in each simulation. These simulations allowed to test the ability of the both DA and JK sources to reduce their throughput when a new user join a shared path, and to increase their window when any user stops. Table 2 shows that both algorithms update the window size of the sources according to the “residual” resources (the background source is not flow controlled). However, we can make the following remarks:

- the bottleneck link is saturated with the use of the DA sources (2000 packets for both sources in 20 seconds), and operated at 93% of its capacity with the JK sources.

- in the "steady state" i.e. two sources without the background, each JK sources systematically has a packet dropped at the bottleneck gateway each 6 seconds. The number of packet dropped increases when the background source joins the network. DA sources, however, reduce their throughput sufficiently early to prevent any packet loss even when the conditions on the network are changing (e.g. a background source joining or leaving the transmission path).

Time	Source	Type	Sent	Dropped	Type	Sent	Dropped
20	A	DA	976	0	JK	905	3
	B	DA	965	0	JK	883	3
	C	Bg	0	0	Bg	0	0
40	A	DA	996	0	JK	927	3
	B	DA	1004	0	JK	930	3
	C	Bg	0	0	Bg	0	0
60	A	DA	467	0	JK	504	7
	B	DA	534	0	JK	508	7
	C	Bg	999	0	Bg	949	44
80	A	DA	462	0	JK	499	7
	B	DA	538	0	JK	463	7
	C	Bg	1000	0	Bg	991	6
100	A	DA	932	0	JK	901	3
	B	DA	1023	0	JK	882	3
	C	Bg	3	0	Bg	12	0
120	A	DA	957	0	JK	917	3
	B	DA	1043	0	JK	928	3
	C	Bg	0	0	Bg	0	0

Table 2: Comparison of the dynamic behavior of both DA and JK algorithms

5 The Delay Algorithm Revisited

5.1 Raising the reference delay

The use of the minimum observed round trip delay as a reference value for the control may have a negative side effect in some cases when the intrinsic propagation delay D_t of the path increases because of route changes e.g. when a failing terrestrial link ($d = 20\,ms$) is replaced by a satellite link ($d = 300\,ms$). This may lead to successive window size reduction as the measured round trip time will be considered too high compared the minimum value. The reference value should then be updated if the network conditions change.

A possible way to solve the problem is to let the reference value to be the minimum of the last M measures of the round trip time. We simulated such dynamic raise of the reference value but we did not obtain satisfactory results: this produced a positive feedback effect, and the "equilibrium point" drifted from the zero queueing at the bottleneck to higher values. This led to frequent packet losses when new users join a loaded link.

Another way to circumvent the problem is to consider that the conditions on the network have changed if the measured delay is too high comparing to the reference delay for MAX-REDUCE consecutive times: it means that we update the reference delay after MAX-REDUCE consecutive

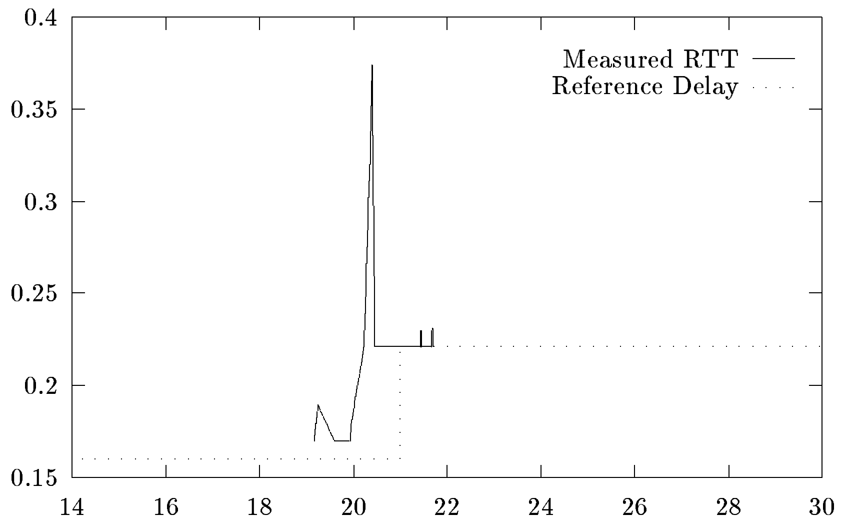

Figure 7: Increase of the delay reference value (in s) after a network conditions change

reduction of the window size. We tested this policy by simulation: we started the simulation with a network configuration corresponding to Figure 2 (with one DA source and $d_{GC} = 50\,\mathrm{ms}$). At $t = 20\,s$ the propagation delay of this link is switched to $d_{GC} = 100\,\mathrm{ms}$. The DA algorithm recovered and the reference delay value was updated after the fourth window reduction (MAX-REDUCE = 4). The results are reported in Figures 7 and 8.

5.2 A shade of aggressivity

The Delay Algorithm uses a "conservative" policy to increase the window: the reference delay value corresponds to "zero queueing" at the bottleneck gateway. Another variant is possible: if the delay increases substantially, the window is reduced to a value that corresponds to a minimal buffer occupation at the gateway and not to "zero queueing". This may be accomplished by adding to the value of congestion window computed according the Delay Algorithm a small increase of 1 or 2 packets corresponding to the minimal buffering at the gateway. This policy avoids a drastic reduction of the window size, even when the demand is increasing. On the other hand, this policy leads to better fairness in the resource sharing: the minimal buffering offset is a more aggressive increase condition which allows to leave a possible unfair stable position. The simulations with a minimal buffering of two packets show that the bandwidth is equally shared by two DA sources even with various small variations of the propagation delay d_{BG}. Figure 9 corresponds to the network configuration depicted in Figure 4, with a delay $d_{BG} = 9.8\,ms$: note the equal share of the bandwidth between the two sources. Figure 5 showed that, with the same network configuration, the JK sources received a completely biased bandwidth allocation.

Another optimistic policy is possible for the window size increase: when the delay is stable, the source probes additional resources by increasing its window. If the delay increases "substantially" (i.e. $D_a^{i+1} > K.D_t^i$), the window will be decreased after a round trip time. This allows

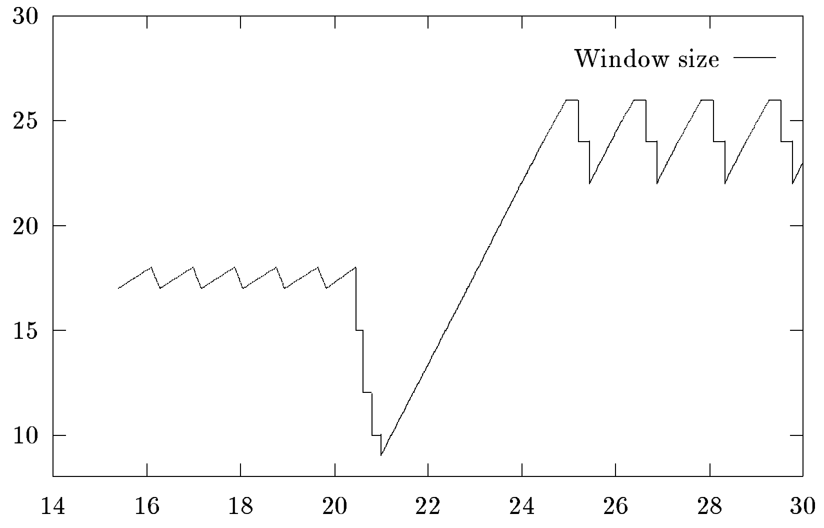

Figure 8: The corresponding window size variation

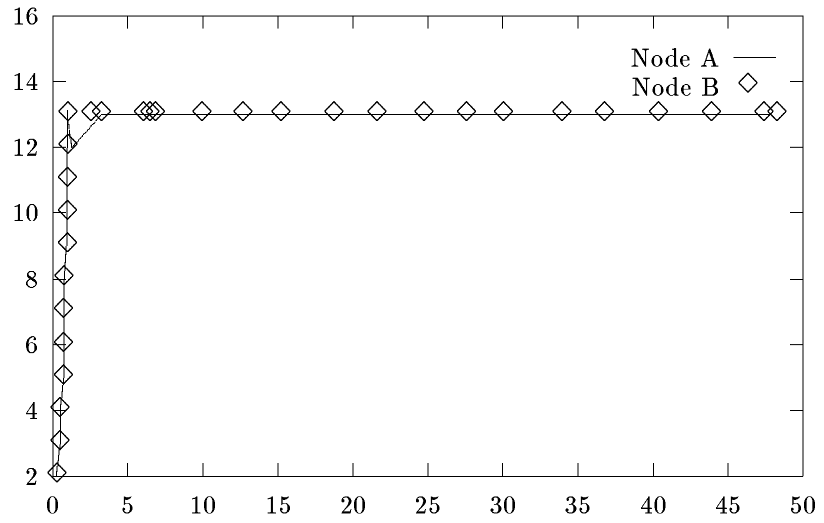

Figure 9: Window size for two DA sources with "Minimal-Buffer = 2"

for a more rapid adaptation to changing network state but may possibly lead to the loss of some packets if the buffer capacity of the bottleneck gateway is reduced. The saw tooths in Figure 8 are due to this optimization. When the demand and the resources are stable, these oscillations cause delay variations. Thus, a tradeoff should be made between "response time" (aggressivity) and "quality of service in the steady state" (amplitude of the oscillations).

6 Fairness issues

The Delay algorithm has also been implemented and tested on the Internet, where hosts implement the standard slow start TCP code. The results of the experiments are detailed in [23]. They show that the delays for the TCP connections follow the same cyclic behavior as the congestion window. We observe that the average round trip delay is much lower for the DA connection than for the TCP connection. Furthermore, the variance of the delays is also lower for the DA connection. However, the difference between the standard deviations for the DA and the TCP connections is not as important as what simulation results from Section 3 could have lead us to believe (see Table 1). Furthermore, we observe almost as many packet losses in the DA and the TCP connections, although we expect packet losses in DA connections to be very infrequent occurrences. These phenomena occur because the experiments were carried out in the Internet, which means that i) traffic from the DA connection competes for network resources with traffic from TCP connections, which does not depend on packet delays, and ii) queueing discipline in the Internet gateways is First-Come-First-Serve (FCFS).

In a FCFS network, queueing delays at a gateway are linked in detail to the arrival patterns of all the conversations in the gateway. Consider for example the situation where a single DA connection shares the resources of a gateway with multiple TCP connections. If any one of the TCP connections sends a large burst of data, then the queueing delays of all other connections increase until the burst has been served. If no packet is lost in the process, the other TCP connection will not react (i.e. will not adjust their window size) to this increase in delays. The DA flow control mechanism will assume that these increased delays mean that the DA connection is causing congestion. However, the DA connection is only affected by congestion caused by other connections. Nevertheless, the flow control mechanism reduces the window size and hence unnecessarily restricts throughput of the DA connection. Thus, we expect that in our experiments, the throughput of a DA connection is lower than the throughput of a TCP connection. This is indeed the case. We observe that the throughput of the DA connection is 30% lower than the throughput of the TCP connection.

We note that this phenomenon would not occur in a so-called rate allocating servers network (RAS network) [14], in which gateways use Fair Queueing or other rate based scheduling disciplines. With FQ discipline, gateways maintain separate queues for packets from each connection. The queues are serviced in a round-robin manner. If a source send packets too quickly, it merely increases the length of its own queue. Thus, the queueing delays experienced by each connection are, to a first approximation, independent of the arrival patterns of other connections. Therefore, if a DA connection observes increasing round trip delays, then this connection is really a source of congestion. As a consequence, we expect the throughput of DA and TCP connections to be equal in a FQ network.

We could not verify the above hypothesis with experiments over the Internet. We used simulations on the simple network configuration shown in Figure 4. In this network, two connections send packets over a common path and share a common gateway. One connection uses Jacobson's algorithm, the other the DA algorithm. Figure 10 shows the evolution of the window sizes for both connections with a FCFS gateway. Figure 11 shows the same evolution with a FQ gateway.

With the FCFS gateway, the throughput of the TCP connection is almost twice as much as the throughput of the DA connection. Furthermore, we observe that the evolutions of the round trip delays follow the same pattern for both connections. This is expected, since packets from the DA connection will be queued behind packets from the TCP connection. This clearly reflects the inability of FCFS gateways to distinguish between different connections and allocate bandwidth and buffer space independently. The situation is quite different with the FQ gateway. There, both connections evenly share the bottleneck capacity. Furthermore, as expected, the average and the standard deviation of packet delays are smaller for the DA connection.

The above results and the simulation results of Section 4 indicate that the DA algorithm can provide connections with low average delay and low delay variance in networks with rate allocating servers. These results confirmed that fairness with end systems congestion control mechanisms cannot be achieved in the presence of aggressive or malicious users without the arbitration of the intermediate gateways.

7 Conclusion

In this paper we studied some issues related to the congestion control problem. The aim of the study was to find an algorithm suitable to be used within the TPX protocol currently under design within the OSI95 ESPRIT project. We proposed to use a delay based congestion control algorithm that eliminates the periodic packet loss experienced with the slow start algorithm and reduces the round trip delay variation. We compared the simulation results of both slow start and the Delay Algorithm and show that the latter allows to eliminate discriminatory behavior of the network due to the traffic "phase effects". Fine adjustments of the Delay Algorithm allow to improve its fairness and guarantee its function on a network with dynamic configuration changes (e.g. re-routing after a link failure). However, this study showed that complete fairness cannot be achieved with FCFS gateways: in this case the aggressive policies get higher share of the available bandwidth. Nevertheless, as the Delay Algorithm allow to reduce round trip delays, it could be used for the support of a video conference application in a network where gateways implement a gateway congestion control mechanism such as the Fair Queueing. The delay algorithm is destined to be used as a combined flow/congestion control mechanism for a video conference application currently under study at INRIA [24]. Future research will investigate application level flow control, in which the video conference application uses the network feedback information, namely packet round trip delays, to adjust the coding rate of the video coder and hence the quality of the images sent over the network.

References

[1] Caceres R., Danzig P., Jamin S., Mitzel D., Characteristics of wide-area TCP/IP conversations, *Proceedings ACM SIGCOMM '91,* Zurich, Switzerland, September 1991.

[2] Cheriton D. R., VMTP: a transport protocol for the next generation of communication systems, *Proceedings ACM SIGCOMM '86,* Stowe, Vermont, pp. 406-415, August 1986.

[3] Chesson G., XTP/PE Design Considerations, in *Protocols for High-Speed Networks,* H. Rudin, R. Williamson, Eds., Elsevier Science Publishers/North-Holland, May 1989.

[4] Clark D., Lambert M., Zhang L., NETBLT: a high throughput transport protocol *Proceedings ACM SIGCOMM '87,* Stowe, Vermont, pp. 353-359, August 1987.

[5] Dabbous W., *Etude des protocoles de contrôle de transmission à haut débit pour les applications multimédias,* PhD Thesis, Université de Paris-Sud, March 1991.

[6] Demers A. Keshav S., and Shenker S., Analysis and Simulation of a Fair Queueing Algorithm, *Proceedings ACM SIGCOMM '89,* pp. 1-12, September 1989.

[7] ESPRIT Project: High performance OSI protocols with multimedia support on HSLAN's and B-ISDN (OSI 95). Technical Annex, September 1990, Internal document, INRIA Sophia Antipolis.

[8] Floyd S., Jacobson V., Traffic Phase Effects in Packet-Switched Gateways, *Computer Communication Review,* Vol. 21, No. 2, pp. 26-42, April 1991.

[9] Gerla M., Kleinrock L., 'Flow control: A comparative survey *IEEE Trans. Comm.*, Vol. 28, April 1980.

[10] Huitema C. A high speed network connection between NSF and INRIA, Research note, INRIA Sophia Antipolis, October 1987.

[11] Jacobson V. Congestion avoidance and control, *Proceedings ACM SIGCOMM '88,* Stanford, California, pp. 314-329, August 1988.

[12] Jain R., A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks, *Computer Communication Review,* Vol. 18, No. 4, pp 56-71, October 1989.

[13] Keshav S., The REAL network simulator, Technical Report UCB/CSD 88/472, University of California at Berkeley, December 1988.

[14] Keshav S., Agrawala A. , Singh S., Design and analysis of a flow control algorithm for a network of rate allocating servers, in *Protocols for High-Speed Networks II*, Elsevier Science Publishers/North-Holland, April 1991.

[15] Mankin A., Random Drop Congestion Control, *Proceedings ACM SIGCOMM '90,* Philadelphia, Pa., pp. 1-7, September 1990.

[16] Mankin A., Ramakrishnan K., Gateway Congestion Control Survey, RFC-1254, August 1991.

[17] Mitra D. Seery J. B., Dynamic adaptive windows for high speed data networks: Theory and simulations, *Proceedings ACM SIGCOMM '90,* Philadelphia, Pa., pp. 30-40, September 1990.

[18] Ramakrishnan K. K., Jain R., A Binary Feedback Scheme for Congestion Avoidance in Computer Networks, *ACM Transactions on Computer Systems,* Vol. 8, No. 2, pp. 158-181, May 1990.

[19] Robinson J., Friedman D,, Steenstrup M., Congestion control in BBN packet-switched networks, *Computer Communication Review,* Vol. 20, No. 1, pp. 30-39, January 1990.

[20] Shenker S., Zhang L., Clark D., Some Observations on the Dynamics of a Congestion Control Algorithm, *Computer Communication Review,* Vol. 20, No. 5, pp. 30-39, October 1990.

[21] Wang Z., Crowcroft J., Eliminating Periodic Packet Losses in the 4.3 Tahoe BSD TCP Congestion Control Algorithm, CCR, Volume 22, Number 2, April 1992.

[22] Zhang H., Keshav S., Comparison of Rate-Based Service Disciplines *Proceedings ACM SIGCOMM '91,* Zurich, Switzerland, September 1991.

[23] Dabbous W., Bolot J., *Study of congestion avoidance mechanisms,* Technical Report, Deliverable INRIA-5, ESPRIT Project OSI95.

[24] Huitema C., Turletti T., *Software Codecs and Workstation Video Conferences* Proceedings of iNET '92, Kobe, Japan, June 15-18, 1992.

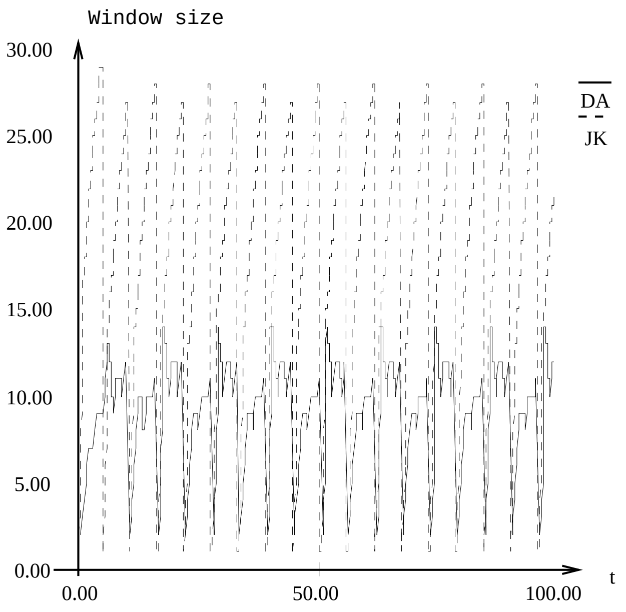

Figure 10: Competition between DA and JK sources: FCFS gateways

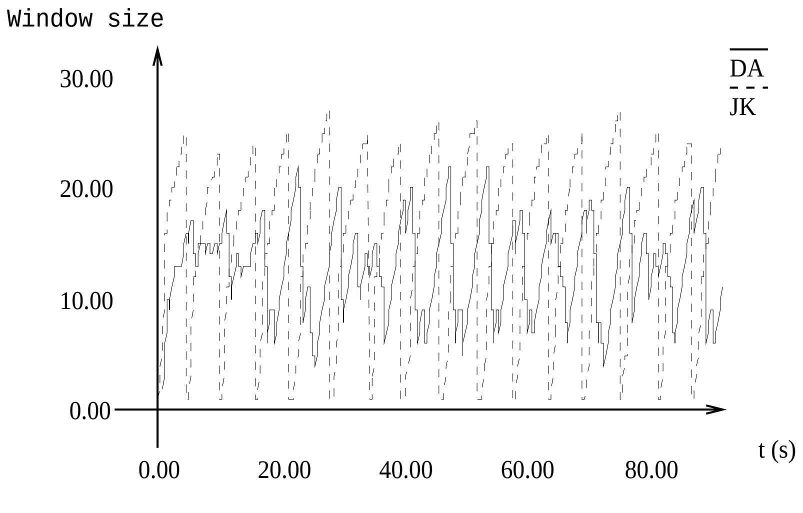

Figure 11: Competition between DA and JK sources: FQ gateways