The interaction of the TCP flow control procedure in end nodes on the proposed flow control mechanism for use in IEEE 802.3 switches

J. Wechta, A. Eberlein, F. Halsall

Department of Electrical & Electronic Engineering

University of Wales, Swansea, UK

eewechta@swansea.ac.uk

Abstract

The existing Ethernet and Token Ring based networks, known as legacy LANs, are no longer able to satisfy the constantly growing demand for more bandwidth and better quality of service. However, switching technology is a solution that seems to be able to cope with further expanding requirements. Introducing switches with new flow control schemes makes the networks faster and more efficient but, at the same time, this can cause other problems. An example is the interaction between the well-established end-to-end TCP flow control and the hop-by-hop switch flow control. The interaction between these two flow control mechanisms is too complex to be solved using theoretical analysis and hence simulation modelling has been used. The advantages and drawbacks of using both flow control schemes firstly independently and secondly together has been investigated and the results are presented and discussed in the paper.

Keywords

End-to-end, hop-by-hop, TCP, flow control, 802.3, switched LANs

1 INTRODUCTION

The advent of multimedia and related applications is resulting in a significant increase in the bandwidth demands on existing local area networks (LANs) [Ban91]. To meet these demands high-speed derivatives of existing LANs have been developed that operate using switching [Hal96], [Mol96], [Kun97]. Because of their origin they are referred to as IEEE 802.3 switches, a schematic diagram of which is shown in Figure 1.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35388-3_42](https://doi.org/10.1007/978-0-387-35388-3_42)

As can be seen a switch operates using a fixed number of input-output ports with an internal bus architecture that enables frames to be passed from multiple input ports to multiple output ports concurrently [Hel97]. Output buffering is provided at each output port to allow for the possibility of multiple frames arriving at different input ports for the same output. However, the amount of buffering is limited [Bra97] and, when a defined limit has been reached, a flow control mechanism is used between switches to allow the congested switch to request the switches that are connected to its input ports to temporarily stop sending any new frames until the congestion has passed. This method is known as backpressure [Omi96], [Yan95]. The switch flow control mechanism is built into the input port of each switch. It uses a combination of XOFF/XON thresholds and XOFF/XON control frames to ensure that no data frames are dropped due to a lack of free memory in the switch [Wan91].

As will be expanded upon in Section 3, the flow control mechanism that has been proposed for use by switches, affects all sources and hence, as will be seen, this has an impact on the flow and congestion control procedure associated with the Transmission Control Protocol (TCP) running in each end system/host. The TCP flow control procedure [Ste95], [Com95], [Vil94] comprises two main elements: a TCP window mechanism and ACK control frames. The integrated self-clocking mechanism used by TCP allows it to regulate the rate of transfer of data packets by allocating the available bandwidth to the individual traffic sources in a controlled and predictable way.

The research reported in this paper has been carried out in order to investigate the interaction between the flow control used by switch and the TCP flow control scheme running in end-stations. The investigation has been carried out using simulation modelling. Each simulation has been run for a significant length of time in order to ensure that the results are reliable. For the research described in this paper the models of TCP version Reno and an 802.3 switching hub with its 802.3x hop-by-hop flow control have been developed in C++ as simulation modules for the BONEs Designer environment.

The motivation for our investigation is described in **Section 2**. Since our interests concentrate on the interaction between both flow control schemes it is necessary to ensure that the conditions for triggering the switch flow control are specified and satisfied. **Section 3** describes this issue. When both flow control mechanisms are

Figure 1: Switch architecture

working together there is a chance that they have a negative influence on each other under certain circumstances. The simulations related to this issue and their results are described in **Section 4**. Discussion of the results obtained and conclusions drawn are presented in **Section 5**.

2 MOTIVATION FOR THE INVESTIGATION

The bases of the flow control mechanism used by switches is shown in Figure 2. It is a simple backpressure hop-by-hop flow control [Ger88], [Jai90], [Kun93] which is designed to cope with the effect of temporarily congested links. When frames arrive at an outgoing link faster than can be forwarded they need to be stored inside the switch and will firstly be stored within the memory inside the switch. Firstly the frames are stored in the output port buffer, then if this buffer becomes full, they are placed into the shared memory and finally, should the shared memory become full they are stored in input port(s) buffer. Hence if overload condition persist, the related input port buffers will eventually overflow [Hac96].

As shown in Figure 2 the switch flow control compares the amount of data stored in the input port buffer (FIFO queue) with the XOFF threshold value (high-water mark [Wan91]). If the XOFF threshold is reached, an XOFF control frame is sent to the adjacent switch (or the source) which in turn stops sending any new data frames. This causes the FIFO queue to release data until the current queue occupancy reaches the XON threshold (low-water mark). At this point, an XON control frame is sent to the switch/host which, in turn, allows the flow of data frames to be resumed. The levels of XON and XOFF thresholds are chosen to ensure the utilisation of affected link remains as high as possible. A problem with this type of flow control is that there is a possibility that a deadlock occurs [Ozv94].

On receipt of an XOFF message a node stops transmitting new data and waits for an XON message before resuming data transfer. If, for some reason, no XON frame arrives the node will remain silent and no more data flows on this link. The solution proposed in the IEEE 802.3x specification is to put a default delay value into the XOFF

Figure 2: Basics of the switch flow control mechanism

flow control frame. If a node receives an XOFF frame and does not receive XON within the delay specified in the XOFF frame it will start sending data again. The choice of a default delay is an interesting issue in itself, but is out of the scope of this paper. As a reliable transport layer protocol, the TCP is responsible for providing an error-free stream of bytes, delivered in the correct sequence [Com95]. Since the loss of data packets is possible, the TCP must perform retransmissions in order to achieve these characteristics.

In addition the TCP running in two communicating end-systems performs an end-to-end flow control on the data flow. The amount of data being sent by a TCP source is restricted by the window mechanism. At the beginning of a TCP connection, the TCP source can only send a small number of data packets. On receipt of the ACK frames for these data packets, the TCP source then increases the window size and sends more data [Ste95]. In this way the TCP window mechanism prevents a large number of data packets being initially sent into the network.

The window opening procedure is shown in Figure 3, and as can be seen, it has three phases :

- rapid increase (P1) known as “slow start”,
- slow increase (P2) known as “congestion avoidance”,
- constant; with the window fully open (P3).

Initially the window size is one, and the source can send just one data packet. On receipt of the ACK frame the window is increased to two packets and two new packets can be sent. For each received ACK frame, the TCP window is increased by one packet size and two new data packets will be sent. Hence, this phase is called the exponential growth zone [Ste95] since the value of the TCP window size increases exponentially [Jac88]: 1, 2, 4, 8, 16 and so on.

When the TCP window reaches half of its maximum size, the slow increase phase is entered. In this phase the growth slows down and increases by only one data packet per round trip time (RTT); that is when all data packets from the current window become acknowledged. This phase ends when the TCP window becomes fully open.

When the TCP window is fully open the TCP connection is in a kind of equilibrium, since as a packet leaves the network a new packet enters. The TCP has a self-clocking

Figure 3: TCP window opening procedure

mechanism which is maintained by ACK frames. The delay experienced by a data packet and its ACK frame is used to dynamically adjust the data flow to both the available bandwidth and the prevailing network loading.

When a TCP data packet is lost, the sender can recover using either the Fast Retransmit scheme [Ste97] or to wait for the retransmission time-out to expire. The Fast Retransmit scheme is triggered on the arrival of at least three duplicate ACK frames which indicates the need for the missing packet to be retransmitted. This retransmission is followed by the Congestion Avoidance procedure [Ste97] which reduces the TCP window size by half. In contrast, if the sender relies on the retransmission time-out, the packet transmission rate is reduced drastically since the retransmission is followed by the Slow Start [Ste97] procedure with the window size reset to 1 packet. As can be seen, in both cases the value of the traffic offered by a sender to the system decreases but, in the second case, the decrease is more drastic.

In this paper the end-to-end TCP and back-pressure hop-by-hop switch control schemes have been treated as a complementary solution. However in many papers (e.g. [Mis92],[Mis96],[Ozv94]) the two flow control schemes have been examined as alternative solutions. The primary motivation for carrying out the research therefore was to investigate the effect of operating both flow control schemes concurrently. To highlight any effects, various combinations of the two flow control schemes being enabled or turned off have been investigated.

3 THE ONSET OF SWITCH FLOW CONTROL

As indicated earlier, there are three types of buffer within a switch: the output port buffer, the shared memory and the input port buffer. This is shown in diagrammatic form in Figure 4. The input port buffer is a buffer dedicated to the end-node connected to this input port and hence has exclusive use of this buffer. The primary task of an input port buffer is to hold received packets awaiting transfer to an output port during periods when short bursts of data packets arrive. The output port buffers are provided to hold packets during the times when the output link is heavily loaded. The shared memory provides additional storage which can be used to hold waiting packets, when destination output port buffer is full, and increases the flexibility of the internal structure of the switch. If there is a congested link on the path between a TCP source and a TCP receiver, the TCP data segments will not be relayed but instead stored within the affected switch. These data segments may be kept in the available memory

Figure 4: The switch memory storage components

storage in each switch along the path [Bol93]. However, the available storage will not be filled randomly. The storage which is the closest to the congested link will be filled first, then the next closest and so on. Hence, the sequence of the storage is filled inside each switch is as follows: first the output port buffer is filled, then the shared memory, and finally the input port buffer. To illustrate the effect of congestion, we assume the three-stage switch topology as shown in Figure 5.

It should be stressed that XOFF/XON traffic will only be present on an input link of a switch that has not enough memory left to store the incoming data packets. Since the TCP window mechanism regulates the amount of data currently in transit, it is helpful to first analyse in more detail the amount of data which is produced by TCP sources, and the available storage in the switches along the path. In practice this was done before the main set of simulations related to flow control interaction were carried out.

TCP restricts the maximum amount of data in transit with the help of the TCP output window. Hence it is possible to determine the maximum overall amount of data currently in transit. Although the maximum TCP window size is usually 64 kB [Ste95], a maximum window size of 128 kB has been chosen for this simulation. Such a large TCP window size was necessary in this simulation in order to fill the switch storage and to trigger the switch flow control. The same simulation results can be obtained by using a larger number of data connections with smaller TCP window size (e.g. 64kB or 32kB) or by using a smaller TCP window size and less storage within the switch. In order to keep the simulation time low the configuration with 10 clients and 128kB window size has been chosen. Since the window size is anticipated to be larger in the future, the TCP header allows the definition of the Window Size Option which supports window sizes up to 1 GB.

We assume that all clients request file transfers from the server. Each client runs a receiver process and establishes a connection to a sending process running on the server. We examined the worst possible case where all data transfers originated at more or less the same time and flowed from server to clients. The following equation describes the relation between the amount of data in transit, the number of TCP sender processes (later on referred to as a source) and the maximum window size:

$$A = N * \text{MaxTCPWindowSize} \quad (1)$$

The following parameter values were used in all equations:

N = number of connected TCP sources, $\text{MaxTCPWindowSize} = 128\text{kB}$,

$\text{OutputPortBufferSize} = 32\text{kB}$, $\text{InputPortBufferSize} = 16\text{kB}$, $\text{SharedMemorySize} = 512\text{kB}$

Figure 5: The examined three-switch topology

The value of A is shown in Figure 6, which indicates the maximum amount of data which is currently in transit when the TCP output window of each source is fully open, that is, 128kB. The points below line A can be interpreted as the possible amount of data in transit if the TCP windows of the sources are not yet fully open. Under congestion conditions the buffers of switch 2 are filled first. The amount of data that may be stored inside Switch 2 is shown in Figure 5 using the equation of line B. The equation for line B is as follows:

$$B = \text{OutputPortBufferSize} + \text{InputPortBufferSize} + \text{SharedMemorySize} \quad (2)$$

Since the equation is not a function of the number of TCP sources, line B in Figure 6 has the constant value of 560kB.

The size of the input port buffer is 16kB. However because of the XOFF/XON thresholds, the occupied space of the Input buffer varies. It can be estimated as the average value of the two extremes, when the input port buffer is full (16kB) and when it is empty (0kB). The average value is thus 8kB. Nevertheless, in order to provide a worst case calculation, the value of 16kB has been chosen, which, in practice, will only occur in rare situations.

If link L3 continues to be congested, the memory buffers in the preceding switches will start to be filled. It should be stressed that the buffers of switch 1 will only be filled when all the buffers of switch 2 have already been filled. Hence the combined memory available is as shown in Figure 6. The equation for line C is as follows:

$$C = B + \text{OutputPortBufferSize} + N * \text{InputPortBufferSize} + \text{SharedMemorySize} \quad (3)$$

Where: B = storage inside switch 2

As can be seen in Figure 6, the amount of memory available (Line C) increases with the number of TCP sources in the system. In summary, therefore, line A represents the amount of data in transit when N TCP sources are sending frames. Line B represents the storage available inside switch 2 and line C represents the amount of storage inside both switches 2 and 1.

Figure 6: Amount of data in transit between multiple TCP source-receiver pairs and the amount of offered storage

Lines B and C divide the amount of data in transit (traffic offered) into three areas:

- below B: traffic offered is smaller than the capacity of the storage of switch 2;
- above B and below C: traffic offered exceeds the storage capacity inside switch 2 but it is still lower than the combined storage capacity of switches 2 and 1;
- above C: traffic offered exceeds the storage capacity of both switches.

The part of line A below line B relates to a topology with 1, 2, 3 or 4 TCP sources, which means that the amount of data produced by them is not sufficient to fill the storage inside switch 2.

The part of line A between lines B and C relates to a topology with 5, 6, 7, 8 or 9 TCP sources. This means that in order to prevent the loss of TCP data packets, XOFF frames need to be generated by the input port of switch 2 and sent to switch 1, that is., XOFF/XON traffic becomes present on link L2 between switches 2 and 1.

The part of line A above line C relates to a topology with 10 or more TCP sources. In this case there is enough traffic offered to fill the storage inside both switch 2 as well as inside switch 1. When the storage inside switch 2 becomes full, XOFF frames will be generated at the input port of this switch and sent to switch 1. When switch 1 receives XOFF frames, it will start to fill its own buffers. When these also become full, all input ports of switch 1 will generate XOFF frames.

Having produced the above graph we are now able to predict when XOFF/XON traffic will be present on any particular link. Furthermore, we can do similar predictions for different parameters such as the amount of shared memory, output port buffer, input port buffer size or the TCP window size. In this case we need to recalculate the equations and derive new graphs. Similar graphs can be derived for different topologies and with varying numbers of switches.

4 SIMULATION RESULTS

The first set of simulations was carried out in order to investigate the effect of enabling and disabling the switch flow control on the throughput and the loss of data packets within the switches. These were carried out for the combined three-switch topology shown earlier in Figure 5. Four different cases were investigated. Some allow for packets being dropped when the storage capacity within a switch is exceeded (switch flow control disabled on selected links) which gives an insight into the TCP retransmission scheme. Others allow us to compare these results with the results of the lossless configurations when the switch flow control is enabled on all links and no data packet loss is expected.

As can be seen in Figure 5, the topology comprises three switches and N TCP sender process / receiver process pairs. The bandwidth of link L1 is 1Gb/s, of link L2 it is 100Mb/s, while the bandwidth of links L3 and L4 is 10Mb/s. The application for all senders is assumed to be the transmission of an infinite file using FTP. The maximum TCP window size is set to the value of 128kB, all links have the same length, the size of the data packets is constant.

The following four cases have been examined:

- 1) switch flow control enabled on all links (i.e., end-node-to-switch links and on switch-to-switch links),
- 2) switch flow control enabled on switch-to-switch links only,
- 3) switch flow control enabled on end-node-to-switch links only,
- 4) switch flow control disabled on all links.

During the investigation the most important parameters quantified were the **throughput** on each link, the **bandwidth** used by a particular sender-receiver pair, and the **number of lost packets**.

4.1 Switch flow control enabled on all links

The *TCP window graphs* in Figure 7(a) show how the size of the TCP window in each end-node changes with time. The graphs show the continuous growth of the size of the TCP windows of 10 sender processes within the server. The growth is over when the TCP window is fully open (128kB). No deviations occur on the graphs which means that no packet retransmissions have been performed by any of the TCP traffic sources. The three phases of the TCP opening procedure (rapid increase, slow increase, and no increase) can be clearly seen.

The *Shared Memory Occupancy graphs* in Figure 7(b) show the rate at which the shared memory is filled in the three switches. The graph contains two curves showing the process of filling the storage inside switch 2 and in switch 1. It can be seen that the storage inside switch 2 is filled quickly. This phenomenon should be seen as a response to the rapid growth of the TCP window and the large number of data packets injected into the system on all links with a bitrate of 100 Mb/s, but the buffers in switch 2 are emptied slower with a bitrate of 10 Mb/s. The process of filling the storage inside switch 1 takes a much longer time (about 20s). This can be explained by each TCP source entering the phase of slow growth of the TCP window and hence a smaller rate of new data packets being injected into the system. No variations in the curves were registered which means that the flow of incoming data packets was constant. Switch 3 is ahead of the bottleneck link L3 and all packets arriving at this switch are immediately relayed to the end-nodes. Therefore no packets are stored in the shared memory within switch 3 and the shared memory occupancy curve for this switch is shown to be 0kB. All graphs show that the system is in a steady state after approximately 30s of simulation time.

4.2 Switch flow control enabled on switch-to-switch links only

The *TCP window size graphs* shown in Figure 8(a) show that the TCP senders experience varying levels of throughput when flow control on the switch-to-end-node links is disabled. This is illustrated by the different times when the window sizes of the senders drop. As can be seen, the ten nodes are divided into the three groups:

- The senders 3, 5 and 8 do not experience a drop of window size. This means that the data packets from them are transmitted first over the slow link resulting in the related ACK frames being received and their corresponding windows carry on increasing.

Scale=10↑3

Figure 7(a): The variation of the TCP window size with time (FC enabled on all links)

Scale=10³

Figure 7(b): The variation of the Shared Memory Occupancy with time (FC enabled on all links)

Indeed, once the data packets from the other sources stop being transmitted, there is an increase in the rate at which these windows increase.

- The senders 1, 2, 6 and 7 have each experienced one retransmission and as a result their window sizes have been divided by two (retransmissions are the result of data packets being discarded in switch 1)
- The senders 4, 9 and 10 have each experienced two retransmissions and as a result their window sizes have been divided twice by two.

The sender numbers have only been assigned in order to relate to the particular senders later on. It needs to be stressed that the packets are dropped at random.

The Shared Memory Occupancy graphs in Figure 8(b) show the effect of data packet retransmissions on the buffer occupancy in the switches. Since the input flow has been slowed down by TCP, but the output remained unchanged, the number of data packets starts to fall. This can be explained by treating the storage inside switch 1 as an extension of the memory inside switch 2. If the line showing the shared memory occupancy inside switch 1 is put above the line showing the memory occupancy inside switch 2, then the line showing the maximum occupancy becomes the minimum occupancy line of switch 1. It can thus be deduced that the whole storage of switch 2 has been emptied and 100KB of data from the storage of switch 1 has been sent before the recovery starts and incoming data starts to fill up the storage again.

The number of retransmitted packets per sender were counted during the simulation:

- senders 3, 5 and 8 experienced no retransmission,
- senders 1, 2, 6 and 7 had one retransmission, and
- senders 4, 9 and 10 have had two retransmissions.

In this case, when switch flow control is enabled on switch-to-switch links only, all TCP output windows were already in the slow increase phase when packets started to get lost. So the effect on a particular sender was relatively small. This is best discussed in relation to the topology which is shown in Figure 9. The storage in switches 2 and 1 were already filled up when the switch flow control was supposed to appear on the switch-to-end node links. However, since the switch flow control was disabled, it was not triggered and hence data packets were lost.

Figure 9 shows the scenario where flow control is enabled on the switch-to-switch links, however disabled on the switch-to-end node links. This means, that after filling the storage in switches 2 and 1, there is no means to send a message to the end node's MAC interface with the request to stop transmitting new data packets. The crosses show the places where data packets are being dropped. The switches in dark grey those switches in which the storage is full before packets are dropped.

4.3 Switch flow control enabled on switch-to-end-node links only

Figure 11 shows the scenario where the switch-to-switch flow control is disabled. After filling the storage in switch 2, it has no means to send a message to switch 1 to request it to stop transmitting new data packets. The cross shows the place where data packets are being dropped. The TCP senders produced only as much data as it was necessary to fill the storage inside switch 2 which means that the senders are still in the phase of

Scale=10↑3

Figure 8(a): The variation of the TCP window size with time (FC enabled on switch-to-switch links only)

Scale=10↑3

Figure 8(b): The variation of the Shared Memory Occupancy with time (FC enabled on switch-to-switch links only)

Figure 9: Flow control disabled on links connecting senders and switch 1

rapid increase of the TCP window size. The switch in dark grey indicates that switch in which the storage is full before packets are dropped.

The *TCP window size graphs* in Figure 10(a) show that all TCP senders are affected by packet losses. In the steady state, all the TCP output window sizes are divided by two. Initially, however, in the case of senders 1, 2, 3 and 4, recovery starts after the initial fall. In the case of sender 5 a second retransmission is performed. For the remaining senders the situation becomes even worse because of the loss of data packets and the need to rely on the retransmission time-out to expire (senders 7, 8, 9 and 10).

The interesting point, however, is that in the steady state all TCP senders, regain the fairness in sharing the bandwidth. The effect of this is a similar window size for different senders even if large variations occurred during the earlier stage. It can also be seen that when the sum of all TCP sender window sizes (the amount of data currently in transit) exceeds the size of storage inside switch 2, packet losses and hence further window size reductions are unavoidable.

The *Shared Memory Occupancy graphs* in Figure 10(b) show some regularity between the emptying and filling of shared memory buffers. However, what needs to be remembered is that each time the storage within the switches becomes full, all new incoming data packets are rejected.

A TCP sender which experiences data packet losses will retransmit the affected data packets. In the examined topologies the highest number of data packet retransmissions occurred in the case when the switch flow control is disabled on the switch-to-switch links. This can be explained by the phase of the TCP window opening procedure. The TCP windows were in the rapid increase phase when packet drops started to occur after filling up the storage in switch 2. Since the flow control on the switch-to-switch links was disabled, incoming data packets were dropped. During the rapid increase of the TCP window more data packets are injected into the system in one burst (2 packets per

Figure 11: Switch flow control disabled on switch-to-switch link between switches 1 and 2

Scale=10¹³

Figure 10(a): The variation of the TCP window size with time (FC enabled on switch-to-end-node links only)

Scale=10³

Figure 10(b): The variation of the Shared Memory Occupancy with time (FC enabled on switch-to-end-node links only)

ACK) which means that the probability of data packet losses from the same source is much higher.

4.4 Switch flow control disabled on all links

This case yields exactly the same results as the case where switch flow control is enabled on the switch-to-end-node links only and disabled on the switch-to-switch links. As just observed, if the switch-to-switch link is run without switch flow control new incoming data packets will be dropped when the storage in switch 2 becomes full. Without the XOFF/XON flow control frames, switch 1 will not be aware of storage problems experienced by switch 2. Hence switch 1 will always forward incoming data packets towards switch 2, even if the latter is not able to store them. Therefore, there is no way to fill the storage in switch 1 and hence the XOFF/XON flow control will not be triggered on the links to the end nodes.

5 DISCUSSION AND CONCLUSIONS

The network resources comprise two components: link bandwidth and buffer capacity within network nodes, e.g., switches. The switch flow control procedure copes with problems related to packets that are already in transit across the network and supports the data packet flow on a hop-by-hop basis. The TCP flow control decides when and how much new data may be injected into the network. Since the TCP is an end-to-end flow control protocol it can be considered as operating on a higher level than the examined switch flow control operating on a hop-by-hop basis.

A TCP traffic source sees the whole set of switches in a route as a long fat pipe. Hence, it does not know that there is a string of switches in-between and it is not aware of any possible problems, which can emerge when, for example, a storage overflow occurs within switches. Clearly, TCP does not have any mechanism which can prevent data packets being dropped due to buffer overflow within switches. The TCP needs a flow control manager on a hop-by-hop level, and the switch flow control performs this role. Concluding, the TCP governs the process of bandwidth sharing, while the switch flow control performs the management of storage within switches. Hence the TCP and switch flow controls operate in different areas and complement each other very well, at least for the analysed topology. However, it is possible to find network topologies and node configurations where using the hop-by-hop flow control can significantly reduce the performance of a network. This phenomenon is described in detail in [Wec98].

Regarding the system response, the TCP flow control is slow, while the switch flow control is fast. There is a delay before the TCP reacts to any indication of congestion. The TCP flow control procedure is based on the timing mechanism maintained by the ACK frames. Since the delay in the examined topology is a function of the number of packets in transit or/and stored in buffers, it is very dynamic and much longer than the packet transmission time over one link. The delay related to switch activities is equal to the transmission delay on a link between two neighbouring switches. Clearly, this is a very short time in comparison to the RTT. However, the actions taken by switches to ensure the smooth flow of data packets through the system remain transparent to the TCP.

Simulation results show that TCP traffic sources, thanks to the window opening procedure, produce much more data than could be transmitted over the network, and hence this data was stored within switches. It should be noted that there is a common belief that long queues are a symptom of congestion. Since such extensive queuing has a negative influence on the end-to-end delay as well as increases the probability of packet losses when new flows are being admitted, one can ask the question whether this effect can be avoided.

The results for the system without switch flow control show that the data flow synchronisation effect [Flo93] is a result of packets being fairly dropped from all flows due to buffer overflow. This is a very harmful effect because it can cause strong load fluctuations in the network and jumping between two temporary states: overloading and underutilising of network resources. Solutions to this problem have been proposed in [Flo93], [Flo97] and can be implemented by introducing a new scheme for dropping packets in congested nodes on a random basis rather than using a simple drop-tail approach. However, combining end-to-end and switch flow control schemes avoids packet dropping since it prevents buffer overflow.

On the other hand dropping packets is considered a proper response from the network but data packet retransmission means wasting valuable bandwidth. Any possible solution which does not waste bandwidth is worth investigating.

6 REFERENCES

- [Ban91] Bandula, W. (1991) High-Speed Local Area Networks and Their Performance: A Survey, *ACM Computing Surveys*, **23**(2), 221-264.
- [Bol93] Bolot, J. (1993) End-to-End Packet Delay and Loss Behaviour in the Internet, *Proceedings ACM SIGCOMM'93*.
- [Bra97] Braden, B., Clark, D. and Crowcroft, J. (1997) *Recommendations on Queue Management and Congestion Avoidance in the Internet*, Internet draft: draft-irtf-e2e-queue-mgt-00.ps.
- [Com95] Comer, D. (1995) *Internetworking with TCP/IP Vol. 1, Principles, Protocols and Architecture*, Prentice Hall.
- [Els96] Elsaadany, A., Singhal, M. and Liu, M.T. (1996) Performance Study of Buffering within Switches in LANs, *Computer Communication*, **19**, 659-667.
- [Flo93] Floyd, S. (1993) Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, **1**(4), 397-413.
- [Flo97] Floyd, S. (1997) *Router Mechanisms to Support End-to-End Congestion Control*, Technical Report, Lawrence Berkeley National Laboratory, <http://ftp.ee.lbl.gov/floyd/red.html>.
- [Ger88] Gerla, M. and Kleinrock, L. (1988) Congestion Control in Interconnected LANs, *IEEE Network*, **2**(1), 72-75.
- [Hac96] Hac, A. (1996) Bandwidth Management and Switch Buffer Allocation in High-Speed Networks with Bursty Traffic, *International Journal of Network Management*, **11**(4), 2-16.

- [Hal96] Halsall, F. (1996) *Data Communications, Computer Networks and Open Systems*, Addison Wesley.
- [Hel97] Held, G. (1997) *High-Speed Networking with LAN Switches*, Wiley Computer Publishing.
- [Jac88] Jacobson, V. (1988) Congestion Avoidance and Control, *Proceedings of ACM Sigcomm'88*, **18**(4), 314-329.
- [Jai90] Jain, R. (1990) Congestion Control in Computer Networks: Issues and Trends, *IEEE Network Magazine*, 24-30, May 1990.
- [Kun93] Kung, H., Morris, R., Charuhas, T. and Lin, D. (1993) Use of Link-by-Link Flow Control in Maximizing ATM Network Performance Simulation Results, *Proceedings of IEEE Hot Interconnects Symposium*, 1-12.
- [Kun97] Kung, H. (1997) *Traffic Management for High-Speed Networks*, Fourth Lecture, International Science Lecture Series, National Academy of Sciences.
- [Mis92] Mishra, P. and Kanakia, H. (1992) A Hop by Hop Rate-Based Congestion Control Scheme, *Proceedings of ACM Sigcomm'92*, 112-123.
- [Mis96] Mishra, P., Kanakia, H. and Tripathi, S. (1996) On Hop-by-Hop Rate Based Congestion Control, *IEEE ACM Transactions on Networking*, **4**(2), 224-239.
- [Mol96] Molle, M. (1996) 100Base-T/IEEE 802.12/ Packet Switching, *IEEE Communications Magazine*, pp. 64-73. August 1996.
- [Omi96] Omidyar, C. and Pujolle, G. (1996) Introduction to Flow and Congestion Control, *IEEE Communication Magazine*, 30-32, November 1996
- [Ozv94] Ozveren, C., Simcoe, R. and Varghese, G. (1994) Reliable and Efficient Hop-by-Hop Flow Control, *Proceedings of ACM Sigcomm'94*, **24**(4), 89-100.
- [Ste95] Stevens, R. (1995) *TCP/IP Illustrated, Vol. 1, The Protocols*, Addison Wesley.
- [Ste97] Stevens, W. (1997) *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery*, RFC 2001.
- [Vil94] Villiamizar, C. (1994) High Performance TCP in ANSNET, *Computer Communication Review*, **24**(5), 45-60.
- [Wan91] Wang, Y. and Sengupta, B. (1991) Performance Analysis of a Feedback Congestion Control Policy under Non-Negligible Propagation Delay, *Proceedings of ACM Sigcomm '91*, 49-57.
- [Wec98] Wechta, J., Eberlein, A., Halsall, F. and Spratt, M. (1998) Simulation-based Analysis of the Interaction of End-to-End and Hop-by-Hop Flow Control Schemes in Packet Switching LANs, *Proceedings of the Fifteenth UK Teletraffic Symposium on Performance Engineering in Information Systems*.
- [Yan95] Yang, C.-Q. and Reddy, A. (1995) A Taxonomy for Congestion Control Algorithms in Packet Switching Networks, *IEEE Network*, 34-45, July/August 1995.