2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

On the Relation Between Congestion Control, Switch Arbitration and Fairness

Ernst Gunnar Gran*, Eitan Zahavi†, Sven-Arne Reinemo*, Tor Skeie*, Gilad Shainer†, Olav Lysne*
**Simula Research Laboratory, Fornebu, Norway. Email: {ernstgr, svenar, tskeie, olavly}@simula.no*
†Mellanox Technologies, Israel/USA. Email: eitan@mellanox.co.il, Shainer@Mellanox.com

***Abstract*—In lossless interconnection networks such as InfiniBand, congestion control (CC) can be an effective mechanism to achieve high performance and good utilization of network resources. The InfiniBand standard describes CC functionality for detecting and resolving congestion, but the design decisions on how to implement this functionallity is left to the hardware designer. One must be cautious when making these design decisions not to introduce fairness problems, as our study shows.**

In this paper we study the relationship between congestion control, switch arbitration, and fairness. Specifically, we look at fairness among different traffic flows arriving at a hot spot switch on different input ports, as CC is turned on. In addition we study the fairness among traffic flows at a switch where some flows are exclusive users of their input ports while other flows are sharing an input port (the parking lot problem).

Our results show that the implementation of congestion control in a switch is vulnerable to unfairness if care is not taken. In detail, we found that a threshold hysteresis of more than one MTU is needed to resolve arbitration unfairness. Furthermore, to fully solve the parking lot problem, proper configuration of the CC parameters are required.

I. Introduction

Traffic congestion in interconnection networks may degrade the network and the compute system performance severely if no countermeasures are taken[1], [2], [3]. Congestion is simply a result of high load of traffic fed into a network link, exceeding the link capacity at that point. Hot spot traffic patterns, network burstiness, re-routing around faulty regions, and conducting link frequency/voltage scaling (lowering the link speed in order to save power), can all lead to congestion. If all these factors are known in advance, the network administrator may alleviate the consequences by effective load balancing of the traffic, but typically this is not the case. Furthermore, in cases where multiple nodes send more data to a single destination than the node can handle, no dynamic re-routing can be done to avoid network congestion. It becomes even more severe when a parallel computer is running multiple different jobs as an on-demand service (e.g. cloud computing), where the resulting traffic pattern becomes totally unpredictable.

Congestion control (CC) as a countermeasure for relieving the consequences of congestion has been widely studied in the literature. In particular, this problem is well understood and solved by dropping network packets in traditional lossy networks such as local area networks (LANs) and wide area networks (WANs). In these environments packet loss and increased latency are indications of network congestion. Herein it is mainly TCP that implements end-to-end congestion control, either by a traditional window control mechanism [4] for detecting dropped packets or through changes in latency [5], [6]. Very often those networks are also over-provisioned in order to avoid congestion.

In high performance computing (HPC) data centers low latency is crucial, and packet dropping and retransmission are not allowed under regular circumstances, contrary to LANs and WANs, due to the loss of performance that is associated with packet drops. Lossless behavior is achieved with credit based link-level flow control, which prevents a node or a switch from transmitting packets if the downstream node or switch lacks buffer space to receive them.

Typically, when congestion occurs in a switch, a congestion tree starts to build up due to the backpressure effect of the link-level flow control. The switch where the congestion starts will be the root of a congestion tree that grows towards the source nodes contributing to the congestion. This effect is known as congestion spreading. The tree grows because buffers fill up through the switches as the switches run out of flow control credits (not necessarily in the root). As the congestion tree grows, it introduces head-of-line (HOL) blocking[7] and slows down packet forwarding that also affects flows which are not contributing to the congestion, severely degrading the entire network performance. The HOL blocked flows become victims of congestion[7].

Congestion control for link-level flow controlled networks cannot be based on a traditional window control mechanism as deployed by TCP, though it effectively limits the amount of buffer space that a flow can occupy in the network[8]. The reason for this is the relatively small bandwidth-delay product in this environment, where even a small window size may saturate the network [7]. A rate control based CC mechanism is more appropriate for link-level flow controlled networks, since it increases the range of control compared to a window based system. The mechanism relies on the switches to detect congestion, and inform the sources that contribute to the congestion that they must reduce their corresponding injection rates. There are basically two ways to inform the source nodes in such an explicit congestion notification scheme. Either the switches can mark the packets contributing to congestion in order to notify the destinations about the situation which subsequently notifies the sources (the forward explicit notification approach), or the switches can themselves generate notification packets that are sent directly to the source nodes (the backward explicit notification approach). The InfiniBand (IB) [9] network applies the former approach, while the emerging Data Center Bridging standard [10] is implementing the latter.

There is a body of work that propose different strategies for congestion notification and marking, e.g. a congested packet can be marked both in the input and output buffer as well as being tagged with information about the severity of the congestion. Furthermore, there are several different approaches to the design of the source response function, i.e. the actions taken to reduce the injection rate, later followed by an increase in the rate when congestion is resolved [8], [11], [12], [13].

There are also congestion control mechanisms targeting link-level flow controlled networks that take a completely different approach. Instead of removing the congestion tree itself, these approaches strive to relieve the unfortunate side effects the congestion tree has on flows not contributing to the congestion. That is, they try to remove the HOL blocking by using special set aside queues for contributors to congestion, effectively making it possible for victim flows to bypass the contributors to congestion without actually removing the congestion tree[14], [15]. Such an approach has the advantage of being able to react immediately and locally at each switch, at the cost of the extra buffers needed for the set aside queues and the added complexity in the switch to manage them. The real cause of the problem, sources injecting too much traffic into the network, is though left untouched. Furthermore, such a CC mechanism is not directly applicable in InfiniBand, the interconnection network technology we will use as a basis for our congestion control and fairness studies in this paper.

InfiniBand was standardized in October 2000 and over the years it has increased its marked share, when referring to the Top500 list [16], to 42% of the HPC market. Furthermore, 4 out of 7 Petaflop systems in the world are using InfiniBand as the system interconnect. Congestion control was added in release 1.2 of the InfiniBand specification and is to some extent based on the work done by Santos et. al. [8].

InfiniBand hardware with support for CC has been available since June 2008 [17], but the firmware required for using CC has just been released. Recently Gran et. al. presented the first experiences with CC in IB hardware, where they showed that the IB CC mechanism effectively resolves congestion and improves fairness by solving the parking lot problem, if the CC parameters are appropriately set[7]. Another significant contribution is the work done by Pfister et. al. [18], where they studied (through simulations) how well IB CC can solve certain hot spot traffic scenarios in fat tree networks.

The IB standard provides some freedom in the implementation of its CC concept; several design decisions are left to the implementer (as standards typically do). Care must be taken, however, regarding the implementation and the use of both the *threshold* and the *marking rate* CC parameters, in order to be able to resolve congestion and achieve fairness. Fairness is an important property of any interconnection network. Different traffic flows, having the same priority, should all get equal access to shared resources in the network. InfiniBand as well as other interconnection networks uses a round robin arbitration scheme[19] for selecting the next packet to be sent over a switch output port.

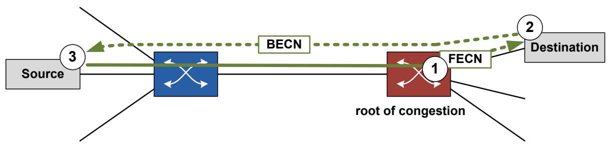


Figure 1. Congestion control in InfiniBand.

In this paper we study the relation between CC, switch arbitration, and fairness. Specifically, we look at fairness at a switch in two types of situations: I) Fairness among different traffic flows arriving at a hot spot switch on different input ports (solving the congestion problem without introducing unfairness), and II) Fairness among traffic flows where some flows are exclusive users of their input ports while other flows are sharing an input port (essentially solving the parking lot problem[20]).

We outline the parameter use and implementation of IB CC around a simulation model. In addition we present results from experiments with hardware to confirm that there is a good correlation between our simulator and the hardware behavior.

The reminder of the paper is organized as follows: Section II gives an overview of the CC mechanism supported by IB. In section III we describe our simulation model, before we in section IV and V study fairness of a IB CC capable switch in the two types of situations explained above. In section VI we show the correlation between our simulator and the hardware, before we conclude in section VII.

II. Congestion Control in InfiniBand

The IB CC mechanism, specified in the InfiniBand Architecture Specification release 1.2.1[9], is based on a closed loop feedback control systems where a switch detecting congestion marks packets contributing to the congestion by setting a specific bit in the packet headers, the *Forward Explicit Congestion Notification* (FECN) bit (fig. 1 (1)). The congestion notification is carried through to the destination by this bit. The destination registers the FECN bit, and returns a packet with the *Backward Explicit Congestion Notification* (BECN) bit set to the source (fig. 1 (2)). The source then temporarily reduces the injection rate to resolve congestion (fig. 1 (3)).

The exact behaviour of the IB CC mechanism depends upon the values of a set of CC parameters governed by a *Congestion Control Manager*. These parameters determine characteristics like when switches detect congestion, at what rate the switches will notify destination nodes using the FECN bit, and how much and for how long a source node contributing to congestion will reduce its injection rate. Appropriately set, these parameters should enable the network to resolve congestion, avoiding HOL blocking, while still utilizing the network resources efficiently.

1) Switch Features: The switches are responsible for detecting congestion and notifying the destination nodes using the FECN bit. A switch detects congestion on a given *port* and a given *Virtual Lane* (Port VL) depending on a $threshold$ parameter. If the threshold is crossed, a port may enter the Port VL congestion state, which again may lead to FECN marking of packets.

The $threshold$, represented by a weight ranging from 0 to 15 in value, is the same for all VLs on a given port, but could be set to a different level for each port. A weight of 0 indicates that no packets should be marked, while the values 1 through 15 represent a uniformly decreasing value of the threshold. That is, a value of 1 indicates a high threshold with high possibility of congestion spreading, caused by Port VLs moving into the congestion state too late. A value of 15 on the other hand indicates a low threshold with a corresponding low possibility of congestion spreading, but at the cost of a higher probability for a Port VL to move into the congestion state even when the switch is not really congested. The exact implementation of the threshold depends on the switch architecture and is left to the designer of the switch. We will in section IV see that precaution needs to be taken not to violate the fairness of the switch when implementing the threshold.

A Port VL may enter the congestion state if the threshold is crossed and it is the root of congestion, i.e. the Port VL has available credits to output data. If the Port VL has no available credits, it is considered to be a victim of congestion and shall not enter the congestion state[1]. When a Port VL is in the congestion state its packets are eligible for FECN marking. A packet will then get the FECN bit set depending on two CC parameters at the switch, the $Packet\_Size$ and the $Marking\_Rate$. Packets with a size smaller than the $Packet\_Size$ will not get the FECN bit set. The $Marking\_Rate$ sets the mean number of eligible packets sent between packets actually being marked. With both the $Packet\_Size$ and the $Marking\_Rate$ set to 0, all packets should get the FECN bit set while a Port VL is in the congestion state.

[1]If the $Victim\_Mask$ is set for the port, then the switch will move the Port VL into the congestion state independently of the number of available credits. The $Victim\_Mask$ is typically set for switch ports connection HCAs to the switch as an HCA will never detect congestion itself.

2) Channel Adapter Features: When a destination CA receives a packet with a FECN bit, the CA should as quickly as possible notify the source of the packet about the congestion. This is done by returning a packet with the BECN bit set back to the source. The packet with the BECN bit could either be an acknowledgement packet (ACK) for a reliable connection or an explicit *congestion notification packet* (CNP). In either case it is important that the ACK or the CNP is sent to the source as soon as possible to ensure a fast response to the congestion.

When a source CA receives a packet with the BECN bit set, the CA lowers the injection rate of the corresponding traffic flow. To determine how much and for how long the injection rate should be reduced, the CA uses a *Congestion Control Table* (CCT) and a set of CC parameters. The CCT holds *injection rate delay* (IRD) values that define the delay between consecutive packets sent by a particular flow (the IRD calculation being relative to the packet length). Each flow with CC activated holds an index into the CCT, the $CCTI$. When a new BECN arrives, the $CCTI$ of the flow is increased by $CCTI\_Increase$. The CCT is usually populated in such a way that a larger index yields a larger IRD. Then consecutive BECNs increase the IRD which again decreases the injection rate. The upper bound of the $CCTI$ is given by $CCTI\_Limit$.

To increase the injection rate again, the CA relies on a $CCTI\_Timer$, maintained separately for each SL of a port. Each time the timer expires, the $CCTI$ is decremented by one for all associated flows. When the $CCTI$ of a flow reaches zero, the flow no longer experience any IRD.

The IB CC can operate either at the Service Level (SL) or at the Queue Pair (QP) level at an HCA. Any lowering of the injection rate as a result of BECN reception, then affects the whole SL or the single QP depending on the level of CC operation. While operating at the SL level may require less resources at the HCA than operating at the QP level, choosing the SL level will have a negative impact on both fairness and performance. The reason is that a single traffic flow contributing to congestion will lower the injection rate of all traffic flows within the same SL at the HCA. This could include traffic flows not contributing to the congestion at all as they are not going through the root of the congestion tree, but headed for other parts of the network. This type of unfairness is avoided if the CC operates at the QP level. We will not consider this type of unfairness any further in this paper, as we focus on the fairness provided by switches running CC.

III. The Simulation Model

Our network simulator and switch model is built on the OMNet++ platform[21]. It is based on the IB model made available to the OMNeT++ community by Mellanox Technologies Ltd in 2007/2008. We have ported this model to the OMNeT++ 4 environment, made several bug fixes, implemented CC support, and added some general extensions to it to make it more suitable for our studies.

Below we give a brief overview of the features and the detail level of our simulation model. A more detailed description of the simulator is given in [22].

A. The IB Model

The IB model consists of a set of modules to simulate an IB network with support for the IB flow control scheme, arbitration over multiple virtual lanes, congestion control, and routing using linear forwarding tables.

The two building blocks for creating networks using the IB model are the *Host Channel Adapter* (HCA) module and the *Switch* module. During a simulation an HCA represents both a traffic generator, traffic injector and a traffic sink in the network, while a *Switch* acts as a forwarding node. The HCA supports several traffic generation schemes, e.g. varying the injection rate, the packet size and the destination node distribution. The *Switch* is modelled as an input buffer architecture with support for virtual lanes, virtual output queuing (VoQ) and virtual cut through switching. It uses round robin arbitration over the different VLs and multiple input ports.

The InfiniBand Congestion Control mechanism is implemented by the *Congestion Control ManaGeR* (CCMGR) module. The CCMGR is part of the HCA and the *Switch*, and manages everything related to congestion control in these modules in compliance with IB CC as described in the previous section.

IV. Fairness - Type I

In the type I situation we look at the fairness among traffic flows arriving at a switch on different input ports, creating congestion as they are headed for the same output port. In an IB CC capable network, the switch will detect congestion as soon as the CC threshold is crossed, and mark the contributing packets to tell the sources about the contention at the switch.

Round robin (RR)[23], [19] is thought to be a fair arbitration scheme for the type I situation. All input ports at a switch are served in a round robin fashion. An input port currently accessing an output port, will not get access to the same output port again until all other input ports requesting the same output port has been granted access. We will use an example to illustrate the fairness of the RR scheme. Figure 2 shows a switch connecting seven end nodes, where all links have the same bandwidth. Now, let us add traffic flows to the network as indicated by the arrows in the figure, and study the throughput of the different flows. The flows are added one by one, with one second intervals, until all five flows are active. The startup sequence follows the numeric ordering, starting with *H1*. Notice that one flow, the one from *H1*, is headed for the node *H4*, while the four other flows are all headed for the same destination, *H5*.

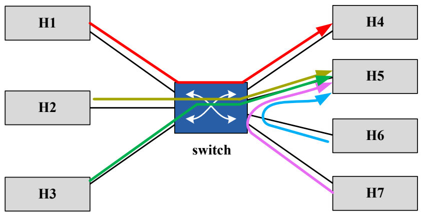


Figure 2. Topology and traffic flows.

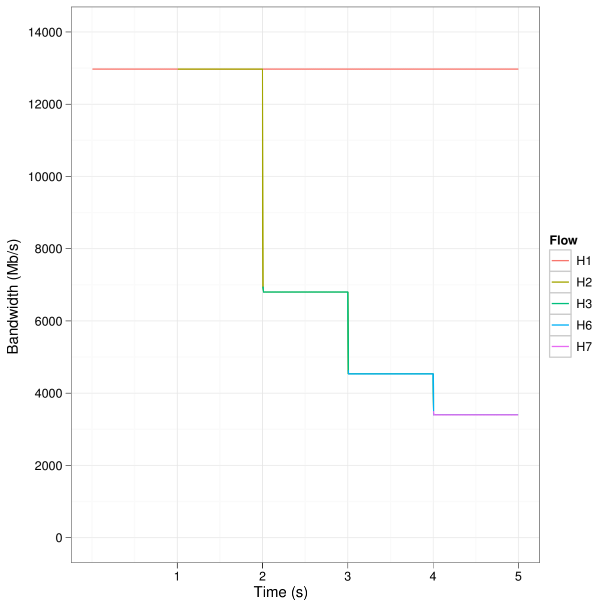


Figure 3. Throughput - No congestion control.

Figure 3 shows the throughput of the five traffic flows during a simulation. The flow from *H1* to *H4* achieves a steady throughput of 13Gbps during the whole simulation. This is as expected as this traffic flow, being the only one headed for *H4*, is independent of all the other traffic flows. The throughput of this flow will serve as a reference as we now add the four flows headed for *H5*. After one second we add the first one, that is, the one originating from *H2*. This flow also maintains a 13Gpbs throughput for one second. Then, at 2s, we add the second traffic flow towards *H5*. Now the two flows headed for *H5* each experience a drop in performance down to half the capacity of the bottleneck link, the link from the switch to *H5*. Each flow is given the same access to this link, that is, the arbitration scheme is fair. As we add the third and fourth flow headed for *H5*, we see a corresponding drop in performance for each new additional flow. However, the available bandwidth at the bottleneck link is evenly shared among the flows, that is, the RR scheme is fair.

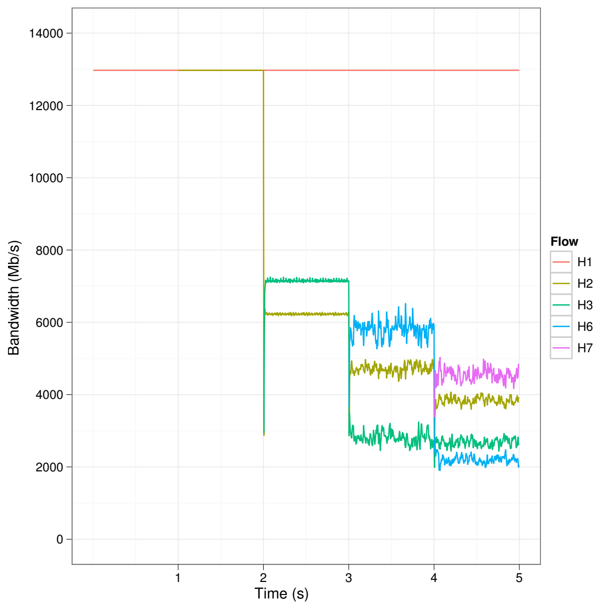


Figure 4. Throughput - Congestion control enabled.

A. Congestion Control Using One Threshold

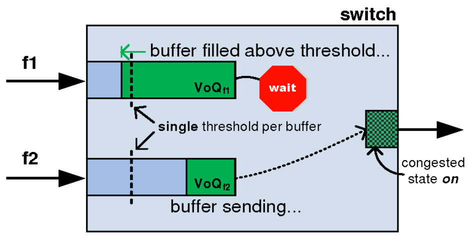
Now, let see what happens to fairness when we enable congestion control (CC) (this small topology could be part of a greater network where it has been found reasonable to use CC). Figure 4 shows the throughput of the five traffic flows running the same simulation as before, but this time with CC turned on. As we can see, even if the RR arbitration scheme is fair, the CC has a negative impact on the fairness. When we at 2s add the flow from *H3*, this flow is able to stabilize at a throughput of a little more than 7Gbps. This is about 1Gbps higher than the other flow going to *H5*, the flow from *H2*. Then, when we add *H6* at 3s we see that the unfairness is even more dramatic. This time the amount of traffic the flow from *H6* is able to get through the bottleneck link is twice as much as what the flow *H3* is able to get through. The flow *H2* settles in between at a little less than 5Gpbs. Enabling CC has introduced unfairness in the network, even if the arbitration scheme in the switch itself is fair. When we add the last flow, *H7*, at 4s the situation continues; the unfairness is still present.

The reason for the observed unfairness in figure 4 has been identified to be the use of a single threshold to detect congestion in the switch. Let us have a closer look at what is happening inside the switch as different traffic flows headed for *H5* are active. First, at any time when only one of the flows is active, no matter how much traffic is sent, the buffers will not fill and no congestion control marking occurs (assuming that all links have the same capacity, and that the end node *H5* is able to remove the traffic from the network as soon as it arrives). When a second flows is active, however, the one flow with the highest injection rate will fill its virtual output queue (VoQ) buffer first[2]. This situation is shown in figure 5a. The figure shows two different input ports of a switch, and the virtual output queues of these two ports, VoQ_{f1} and VoQ_{f2}, corresponding to the shown output port (the one connecting *H5* to the switch in our scenario). The threshold is shown as a vertical dotted line. Here *f1* is the flow with the highest injection rate, and hence the flow filling its VoQ the fastest. Naturally, the fill ratio of VoQ_{f1} crosses the threshold at a time where the other flow is given access to the output port. The crossing of the threshold moves the output port into the congested state, and by that, packets being forwarded on this output port will be marked as contributing to congestion. While in this state, the RR arbitration in the switch will alternate between giving the two flows access to the output port, resulting in each flow experiencing approximately the same amount of congestion control marking. The two flows will lower their injection rates correspondingly. As the flow *f1* initially was the most aggressive flow, and both flows receives approximately the same amount of congestion control feedback, the flow *f2* is actually likely to empty VoQ_{f2} before the fill ratio of VoQ_{f1} is below the threshold. When this happens, *f1* is given sole access to the output port. Now having an injection rate lower than the maximum capacity of the link, the VoQ_{f1} will be emptied below the single threshold and the output port will be moved out of the congestion state, as figure 5b shows.

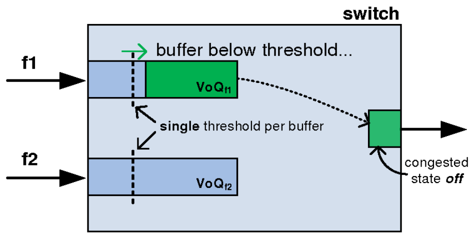
All in all, the situation is somewhat controlled by the most aggressive flow, *f1*. Exactly how many packets that will be marked as contributing to congestion from each of the two flows *f1* and *f2*, depends on the traffic flow characteristics, packet and buffer sizes, and the threshold chosen. Our simulation studies show, however, that in the common situation the different flows receive approximately the same amount of congestion control information. Keeping this in mind, also notice that two different traffic flows governed by congestion control might very well stabilize at different injection rate levels, even if they receive the same amount of congestion control information. This could happen if one of the flows, when congestion occurs, is initially sending at a higher injection rate than the other one. Then, as both flows are throttled by the same amount of congestion control information, the net result is that the two flows are stabilizing at different injection rates after the throttling as well.

Now, look at figure 4 again. Each time we add a new flow contributing to congestion, the new flow starts out with a higher injection rate than the already contributing flows. This is exactly the situation explained in the previous section, and the result is quite visible in the figure: the flow given the largest share of the bottleneck link is always the last flow added - a flow that started its injection at full bandwidth.

[2]In general, different end nodes may very well inject traffic into the network at different injection rates.

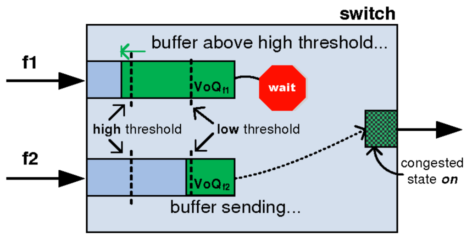


(a) Output port moved into congested state.

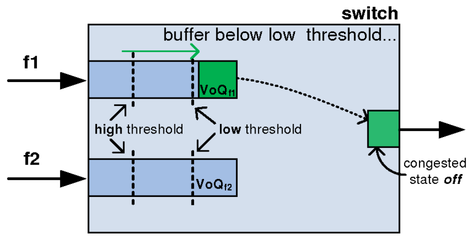


(b) Output port released from congested state.

Figure 5. Congestion control using a single threshold.



(a) Output port moved into congested state.



(b) Output port released from congested state.

Figure 6. Congestion control using two thresholds.

More generally, an aggressive flow always being the last one to empty its VoQ below the threshold, could benefit from this behavior by constantly getting more than its fair share of access to the bottleneck link. This is the behavior observed in figure 4. One way to avoid this unfortunate situation is to introduce a second threshold.

B. *Congestion Control Using Two Thresholds*

Let us now study the situation in figure 2 again, this time with two thresholds per VoQ. Introducing a second threshold, each VoQ now has a lower and upper threshold, as shown in figure 6. When the fill ratio climbs above the high threshold the corresponding output port is moved into the congested state, as shown in figure 6a. The output port is then not released from this state until the fill ratio is lower than the low threshold, figure 6b. By using these two thresholds we ensure that congestion control information continue to be sent for an extended period of time, even after the fill ratio is below the upper threshold. It is no longer possible for a flow to first trigger congestion marking, and then immediately release congestion again as soon as it itself is being granted access to the output port. In particular, an aggressive flow will experience congestion control marking for an extended period of time, even when it has sole access to the output port.

Figure 7 shows the same simulation as before, this time using two thresholds instead of one. As we can see from the figure, fairness is now restored. Each time a new contributor to congestion is added, all contributors quickly settle for their fair share of the congested link, even if the newly added contributor initially was injecting traffic at a much higher rate than the others. The use of two thresholds ensures that any aggressive flow triggering congestion at a VoQ in a switch, will receive a correspondingly high amount of congestion control information, that is, BECNs.

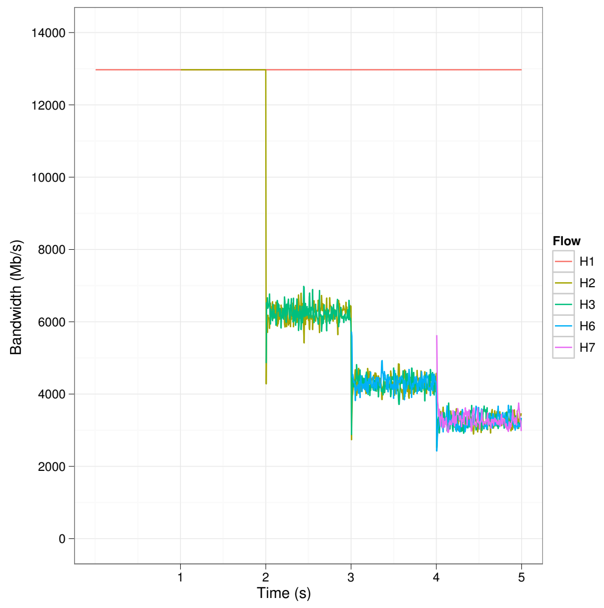


Figure 7. Throughput - Congestion control using two thresholds.

Simulations have shown that an hysteresis of more than one MTU (maximum transmission unit) is needed to resolve unfairness. The distance between the upper and lower threshold used during the simulations shown in figure 7 was a little

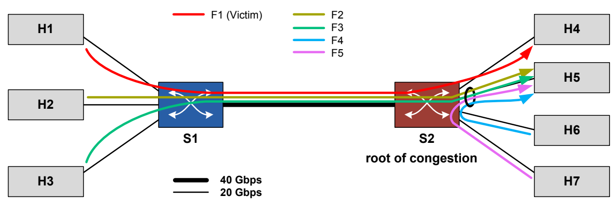


Figure 8. Topology and traffic flows (II).

more than 3 MTUs.

C. A common set of thresholds

For simplicity, each VoQ in figure 6 is shown to have its own set of thresholds. Then CC is triggered as soon as the threshold is crossed in at least one of the VoQs. Another approach is to use a common threshold for all VoQs related to a given output port. Then this common threshold is compared against the sum of the buffer occupancy of all corresponding VoQs. Using a common threshold, there is no differentiation between a single flow occupying a large part of its VoQ, and by that solely triggering congestion marking, and several flows filling smaller parts of their VoQ, but together occupying enough buffer space to trigger congestion marking. Our simulations shows that the behavior of the two approaches are quite similar. What matters is that the threshold parameter is mapped to a fill ratio of the VoQ(s) that ensures fast and proper congestion detection. Special care must be taken when implementing the common threshold approach, to make sure that the threshold maps to a low enough fill ratio to allow for one single VoQ to trigger congestion by its own. This is important as a single flow or VoQ could create congestion alone in certain situations. E.g. a source node injecting traffic into the network faster than the destination node can handle, could singlehandedly create a congestion tree with the root of the tree at the last switch prior to the destination. A single VoQ could also very easily create congestion alone in a network where different links have different bandwidths. We will see an example of this in the next section.

V. Fairness - Type II

Let us turn our attention towards the type II situation. In this scenario some traffic flows are sharing an input port, while other flows are the exclusive users of their input ports. By extending our topology from figure 2 with a second switch, we can exemplify such a situation by adding the traffic flows shown in figure 8. Notice that the switch to switch link has twice the capacity of the other links. Now, at the switch $S2$, the two flows from $H2$ and $H3$ headed for $H5$ are sharing an input port and the VoQ corresponding to the link towards $H5$, while the two traffic flows from $H6$ and $H7$ are exclusive users of their input ports. The Round Robin arbitration scheme provides each input port with the same fraction of the output bandwidth, resulting in an unfair arbitration for the flows heading to $H5$. The two flows from $H2$ and $H3$ will by $S2$ be seen as one flow since they share the same VoQ. Then, implementing RR arbitration, the switch $S2$ will grant access to the link towards $H5$ as if there were only three traffic flows requesting access to it instead of four. The result is that the flows from $H6$ and $H7$ each get 1/3 of the total bandwidth of the link between $S2$ and $H5$, while the last 1/3 is shared by the two flows from $H2$ and $H3$ giving them 1/6 of the bandwidth each. This unfairness, due to the sharing of input ports and buffers by *some* flows at an RR based switch, is often referred to as *the parking lot problem*[24], [20].

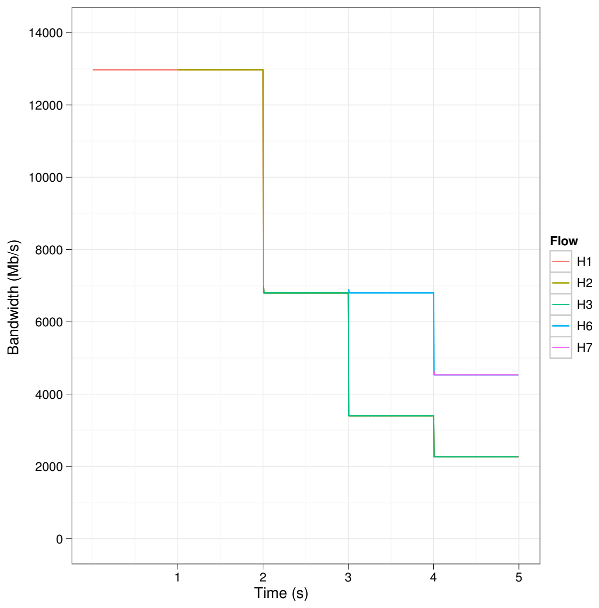


Figure 9. Throughput - No congestion control (II).

Figure 9 shows the throughput achieved by the different traffic flows when simulating the situation of figure 8. As before, a new traffic flow is added to the network each second, starting with the flow from $H1$. The unfairness caused by the parking lot problem is clearly visible as soon as we add the flows from $H6$ and $H7$ at 3s and 4s respectively. During the time period from 3s to 4s, $H6$ achieves the same throughout as $H2$ and $H3$ combined. When $H7$ is added at the time 4s, both $H6$ and $H7$ achieves twice the throughput of $H2$ and $H3$. The parking lot problem is evident. Notice also that HOL blocking is present. As soon as we add the flow from $H3$, a congestion tree builds from $S2$ through $S1$ towards the sources, and the flow from $H1$ is not able to progress any faster than the contributors to congestion $H2$ and $H3$[7].

Most interconnection networks like InfiniBand are using Round Robin arbitration between inputs of the switch. The worst possible unfairness happens when all nodes send

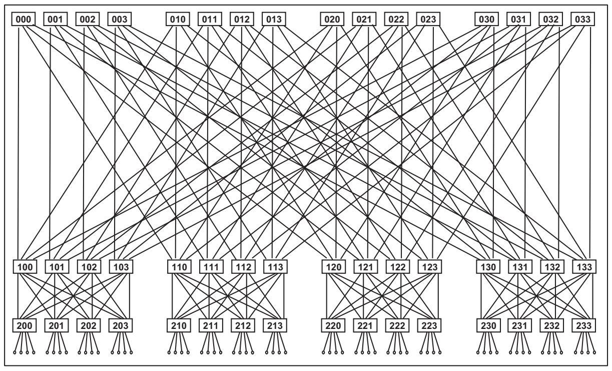


Figure 10. An example 4-ary 3-tree.

data to a single node. Formally, assuming traffic is flowing through M out of the $N-1$ input ports towards an output port, the RR arbitration will provide $1/M$ of the output bandwidth to each input. In the single switch scenario with a network diameter of two, this is fair. In a larger network, however, the M_{s_i} flows from a switch S_i sharing the output port towards the switch S_j, will at S_j together only be assigned $1/M_{s_j}$ of the capacity of the next output port (given that they are all headed for the same output port at S_j as well). That is, in the worst case scenario the unfairness increases by M_{s_x} for each additional switch on the path from the source towards the destination. For large MIN networks where M approaches $N-1$, the worst possible unfairness in bandwidth allocation is: $(N-1)^{D-2}$ where D is the network diameter. For network topologies like k-ary n-trees[25], the exact ratio depends on the routing used. For D-Mod-K routing [26], [27], each down link in the network carries traffic to a single destination. In the example 4-ary 3-tree provided in figure 10, there are 3 first level neighbors to the destination, $4^2-4=12$ second level neighbors and $4^3-4^2=48$ third level neighbors. With RR arbitration, the bandwidth fraction provided is $1/4$ and $1/4/4/4=1/64$ for the first and second level neighbors, respectively. For the third level neighbors, as they all go through the same spine port, the bandwidth fraction for each source is $1/4/4/48=1/768$. More formally, for a k-ary n-tree the furthest sources will in the worst case scenario receive only $1/k^{n-1}/(k^n-k^{n-1})=1/(k^{2n-1}-k^{2n-2})$ of the bandwidth of the last link to the destination, while a first level neighbor will receive $1/k$. It is imminent that with high radix switches the unfairness is so high that it may actually cause transport timeouts in cases of many to one communication. Similarly, timeouts could happen due to unfairness in a network even with low radix switches if the diameter is high.

Some alternatives to RR arbitration, that could solve the demonstrated unfairness, rely on the number of different flows through an input port as a weight for the arbitration.

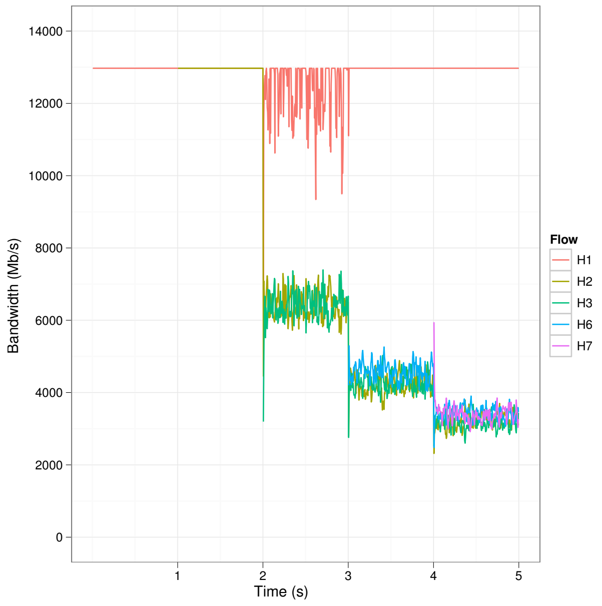


Figure 11. Throughput - Congestion control turned on (II).

The incurred cost of such a solution presented by [28] makes it impractical for Interconnection Networks - where the number of different flows is $O(N^2)$.

The topology in figure 8 is suitable for studying the characteristics of CC in a controlled manner. At the same time, recognize that this topology could be part of a larger topology similar to the 4-ary 3-tree in figure 10. If we now rerun our simulation of the scenario shown in figure 8 with CC enabled, we get the results shown in figure 11. The four traffic flows contributing to congestion are now equalized. The HOL blocking is gone and the parking lot problem has been resolved. Recall from section IV that any aggressive flow will experience more congestion marking than less aggressive flows, given that the CC mechanism is properly implemented using hysteresis, that is, two thresholds. Then, with the introduction of CC, using a marking rate of 1[3], the close neighbors are punished as they are able to pass more traffic through the congested port. The result is that, on average, all flows will be throttled to provide the same bandwidth, as can be seen from figure 11.

While the introduction of IB CC has the potential of solving the parking lot problem, the degree of success in solving this problem is not only related to the hardware implementation of the IB CC, but also the values chosen for the IB CC parameters. E.g. choosing an unfortunate marking rate or $CCTI\_Timer$ could result in the contributors to congestion being slow in settling for their fair share of the bottleneck link. An example is given in figure 12. Here

[3]The marking rate rule applied marks every packet going through a congested link with probability $P(1/(marking\_rate+1))$. This adheres to the IB CC specification.

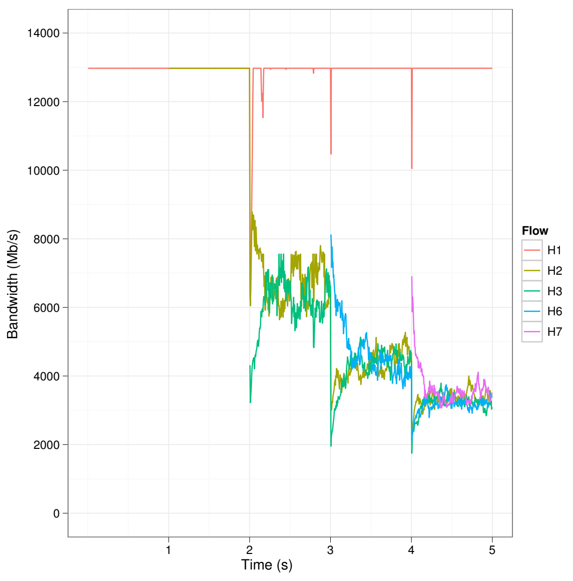


Figure 12. Throughput - marking rate 10.

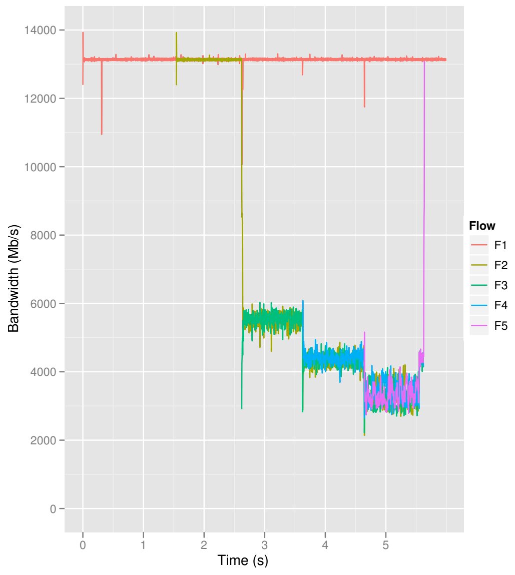


Figure 13. Throughput from hardware - Congestion control turned on.

the marking rate is set to 10, while the $CCTI\_Timer$ is set 10 times as high as before to compensate for the new marking rate. While the parking lot problem is still to some degree solved, an unfairness is present in the network for an extended period of time each time a new contributor is added. How to set the IB CC parameters, possibly independent of topology and traffic patterns, is a subject of ongoing research. Both simulation studies and hardware experiments[7] indicate, however, that the marking rate should be kept low, preferable at 1, to ensure high utilization of network resources and fairness.

VI. A Comparison with Hardware Experiments

In the studies we have presented in this paper we have been using a simulation model to examine the relation between CC, arbitration and fairness. The use of a simulation model was necessary because no hardware that we know of gives us the same flexibility as simulations when it comes to changing the internal behavior of the switch, as done in our study. When using simulations, however, it is important to validate, to the extent possible, the correctness of the model with its real world equvivalent. We have done a general validation of our simulation model against real hardware as documented in[22]. Furthermore, we have compared the simulation results from the threshold implementation using hysteresis (two common threshold values for all VoQs corresponding to a given output port), to hardware experiments using IB CC capable hardware from Mellanox Technologies and Sun Microsystems, now Oracle. A hardware test bed corresponding to the topology showed in figure 8 was put together using two Mellanox InfiniScale IV switches and seven Mellanox ConnectX Host Channel Adapters equipped in seven Sun Fire X2200 M2 hosts. Figure 13 shows the hardware experiment results corresponding to the simulation results shown in figure 11. As the figures show there is a strong correlation between our simulation results and the hardware results, which confirms our confidence in the simulation model.

VII. Conclusions

Switch arbitration and its relation to fair utilization of link bandwidth has been studied for decades. Congestion control in interconnection networks on the other hand is far less understood. This is particularly true for its relation to various fairness aspects.

Ideally, the local switch-fairness provided by well functioning switch arbitration should work independently of the more global stream fairness provided by end-to-end congestion control. In this paper we have demonstrated and explained why this is not always the case. Through simulations calibrated with hardware measurements, we have shown that a straightforward implementation of IB congestion control leads to unfairness even in a simple one switch scenario. Furthermore this unfairness is unstable in the sense that the distribution of bandwidth to the different flows depends on the utilization of the different flows that just happened to prevail at the instance of time when congestion occurred.

We have demonstrated that a good solution to the above problem is to introduce two congestion control marking thresholds. We have also shown that this technique solves the parking lot problem[20] and is quite robust with regards to parameter settings. The detailed relation between our results and CC-parameters settings is still under investigation. Preliminary results indicate that our main results prevail, but that the marking rate influences the speed by which the system converges to a stable state after a change of traffic pattern. A full investigation of this is, however, left for future work.

References

[1] G. F. Pfister and V. A. Norton, “”Hot Spot” contention and combining in multistage interconnection networks,” *IEEE Trans. Computers*, vol. 34, no. 10, pp. 943–948, 1985.

[2] W. Dally, “Virtual-channel flow control,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, 1992.

[3] P. García, J. Flich, J. Duato, I. Johnson, F. Quiles, and F.Naven, “Dynamic evolution of congestion trees: Analysis and impact on switch architecture,” in *High Performance Embedded Architectures and Compilers*, 2005, pp. 266–285.

[4] V. Jacobson, “Congestion avoidance and control,” in *SIGCOMM*. ACM, 1988, pp. 314–329.

[5] L. S. Brakmo and L. L. Peterson, “TCP vegas: End to end congestion avoidance on a global internet,” *IEEE Journal on selected Areas in communications*, vol. 13, pp. 1465–1480, 1995.

[6] C. Parsa and J. Garcia-Luna-Aceves, “Improving TCP congestion control over internets with heterogeneous transmission media,” in *7th International Conferance on Network Protocols (ICNP99)*. IEEE Computer Society, 1999, pp. 213–221.

[7] E. Gran, M. Eimot, S.-A. Reinemo, T. Skeie, O. Lysne, L. Huse, and G. Shainer, “First experiences with congestion control in InfiniBand hardware,” in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, 2010, pp. 1–12.

[8] J. R. Santos, Y. Turner, and G. J. Janakiraman, “End-to-end congestion control for InfiniBand,” in *INFOCOM*, 2003.

[9] *Infiniband architecture specification*, 1st ed., InfiniBand Trade Association, November 2007.

[10] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks - Amendment: 10: Congestion Notification.*, IEEE 802.1Qau-2010 ed., IEEE 802 LAN/MAN Standards Committee, 2010. [Online]. Available: http://www.ieee802.org/1

[11] J. R. Santos, Y. Turner, and G. J. Janakiraman, “Evaluation of congestion detection mechanisms for InfiniBand switches,” in *IEEE GLOBECOM – High-Speed Networks Symposium*, 2002.

[12] J.-L. Ferrer, E. Baydal, A. Robles, P. López, and J. Duato, “Congestion management in MINs through marked and validated packets,” in *PDP*, 2007, pp. 254–261.

[13] ——, “On the influence of the packet marking and injection control schemes in congestion management for MINs,” in *Euro-Par*, 2008, pp. 930–939.

[14] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo, “A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks,” in *HPCA ’05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 108–119.

[15] J. Escudero-Sahuquillo, P. García, F. Quiles, J. Flich, and J. Duato, “FBICM: Efficient congestion management for high-performance networks using distributed deterministic routing,” in *High Performance Computing - HiPC 2008*, ser. Lecture Notes in Computer Science.

[16] “Top 500 supercomputer sites,” http://top500.org/, Nov. 2010.

[17] Mellanox Technologies, “Mellanox announces availability of industry’s first 40Gb/s InfiniBand switch silicon device and product reference platforms,” Press release, June 2008, http://www.mellanox.com/content/pages.php?pg=press_release_item&rec_id=214.

[18] G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Ni, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato, “Solving hot spot contention using InfiniBand architecture congestion control,” Invited paper in High Performance Interconnects for Distributed Computing, july 2005.

[19] N. McKeown, “The iSLIP scheduling algorithm for input-queued switches,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, Apr. 1999.

[20] W. J. Dally and B. Towles, *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004, ch. 15.4.1, pp. 294–295.

[21] “Omnet++ network simulation framework,” http://www.omnetpp.org/.

[22] E. G. Gran and S.-A. Reinemo, “InfiniBand congestion control, modelling and validation.” OMNeT++ 2011, Barcelona, Spain.

[23] W. J. Dally and B. Towles, *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[24] M. Galles, “Spider: A high-speed network interconnect,” *IEEE Micro*, vol. 17, no. 1, pp. 34–39, 1997.

[25] S. Ohring, M. Ibel, S. Das, and M. Kumar, “On generalized fat trees,” in *Parallel Processing Symposium, 1995. Proceedings., 9th International*, Apr. 1995, pp. 37–44.

[26] C. Gomez, F. Gilabert, M. Gomez, P. Lopez, and J. Duato, “Deterministic versus adaptive routing in fat-trees,” in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, 2007, pp. 1 –8.

[27] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, “Optimized InfiniBand fat-tree routing for shift all-to-all communication patterns,” in Concurrency and Computation: Practice and Experience, vol. 22, no.2, pp. 217 –231, 2009.

[28] P. Gratz, B. Grot, and S. Keckler, “Regional congestion awareness for load balance in networks-on-chip,” in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, 2008, pp. 203 –214.