2012 IEEE 26th International Parallel and Distributed Processing Symposium

Exploring the Scope of the InfiniBand Congestion Control Mechanism

Ernst Gunnar Gran, Sven-Arne Reinemo,
Olav Lysne, Tor Skeie
Simula Research Laboratory
Fornebu, Norway
Email: {ernstgr, svenar, olavly, tskeie}@simula.no

Eitan Zahavi, Gilad Shainer
Mellanox Technologies
Israel/USA
Email: eitan@mellanox.co.il,
Shainer@Mellanox.com

***Abstract*—In a lossless interconnection network, network congestion needs to be detected and resolved to ensure high performance and good utilization of network resources at high network load. If no countermeasure is taken, congestion at a node in the network will stimulate the growth of a congestion tree that not only affects contributors to congestion, but also other traffic flows in the network. Left untouched, the congestion tree will block traffic flows, lead to underutilization of network resources and result in a severe drop in network performance.**

The InfiniBand standard specifies a congestion control (CC) mechanism to detect and resolve congestion before a congestion tree is able to grow and, by that, hamper the network performance. The InfiniBand CC mechanism includes a rich set of parameters that can be tuned in order to achieve effective CC. Even though it has been shown that the CC mechanism, properly tuned, is able to improve both throughput and fairness in an interconnection network, it has been questioned whether the mechanism is fast enough to keep up with dynamic network traffic, and if a given set of parameter values for a topology is robust when it comes to different traffic patterns, or if the parameters need to be tuned depending on the applications in use. In this paper we address both these questions. Using the three-stage fat-tree topology from the Sun Datacenter InfiniBand Switch 648 as a basis, and a simulator tuned against CC capable InfiniBand hardware, we conduct a systematic study of the efficiency of the InfiniBand CC mechanism as the network traffic becomes increasingly more dynamic.

Our studies show that the InfiniBand CC, even when using a single set of parameter values, performs very well as the traffic patterns becomes increasingly more dynamic, outperforming a network without CC in all cases. Our results show throughput increases varying from a few percent, to a seventeen-fold increase.

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 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 in a network with link-level flow control, a congestion tree starts to build up due to the backpressure effect of the flow control mechanism. 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

1530-2075/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPS.2012.104

cannot be based on a traditional window control mechanism as deployed by TCP, even 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 [9] architecture applies the former approach[1], 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 left untouched.

Adaptive routing (AR) could be used as a mechanism to alleviate congestion by spreading the traffic in the network onto otherwise idle resources when congestion occurs. Notice, though, that AR as a sole solution to congestion might not be effective. When there is no possible route around an area of congestion (e.g. end node congestion), trying to reroute around the problem will only make the branches of the congestion tree spread out and cause more HOL blocking than when using deterministic routing. To be effective AR needs to work in conjunction with another CC mechanism. Furthermore, the InfiniBand specification does not yet support AR.

InfiniBand (IB) was standardized in October 2000 and over the years it has increased its market share, when referring to the Top500 list [16], to 42% of the HPC market. If we only look at the top 100 supercomputers in the world, the number increases to 62%, while 5 out of 10 Petaflop systems in the world are using IB as the system interconnect. The vast majority of the IB installations are fat-trees with the most notable exceptions being Pleiades (11D hypercube) and RedSky (3D torus).

In 2010 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] and the switch arbitration properly designed [17]. 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 a large degree of freedom, but little guidance, when it comes to configuring the CC mechanism. Care must be taken when configuring the CC parameters because a bad configuration can result in low performance and instability in the network [7].

In this paper we explore the scope of the IB CC mechanism with regards to the robustness and performance of a given parameter set when the communication pattern gradually changes from a static to a dynamic scenario. We describe a set of communication patterns designed to stress the IB CC mechanism and we present a large set of simulation results showing that the IB CC mechanism is both robust and increases performance in all the scenarios studied independent of the communication pattern.

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 different types of congestion trees and the communication patterns used in our simulations.Then the simulation model is described in section IV, before we in section V discuss the simulation results. Finally, we conclude in section VI.

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

[1]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].

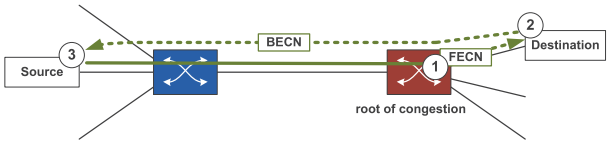


Figure 1. Congestion control in InfiniBand.

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.

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[2]. 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.

2) Channel Adapter Features: When a destination CA receives a packet with a FECN bit set, 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. 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

[2]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.

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. In this paper we only consider CC operating at the QP level.

III. Congestion Trees

When contention leads to congestion in a lossless interconnection network, a congestion tree will form, as explained in section I. Branches of the tree will grow from the root of the tree, backwards towards the sources contributing to congestion, as traffic competing for the contested resources at the root starts to pile up. Buffers and links will be occupied along each branch by traffic headed for the root of the tree. Eventually, a branch might grow all the way back into a source node. The HOL blocking caused by the branches not only results in underutilization of network resources, but will also stimulate further growth of the tree towards nodes initially being victims of congestion. The role of a throttling based congestion control mechanism like IB CC, is to prune the branches of the congestion tree just enough to remove any HOL blocking, while still utilizing the bottleneck resources at the root of the tree.

Exactly how all the branches of a congestion tree grow and get pruned, and by that how the congestion tree itself develops and how much HOL blocking it causes, depends not only on the actions taken by the congestion control mechanism, but obviously also to a great extend on the traffic patterns in the network. A highly dynamic traffic pattern with lively contributors to congestion could lead to a vastly dynamic congestion tree, which again could pose a great challenge to the congestion control mechanism, compared to a more stable tree having a permanent set of contributors. At the same time, a dynamic traffic pattern could by its own dynamic nature, relieve the effects of congestion as the traffic pattern itself will hinder the formation of large congestion trees and any extensive HOL blocking. To be able to conduct a systematic study of the effectiveness of the IB CC mechanism under different traffic scenarios, we start by dividing congestion trees into three nonexclusive categories, based on the nature of their (main) contributors and how dynamic they are. Starting with the least dynamic category, the congestion trees can be *silent*, *windy*, or even *moving* while wind is blowing through their crowns.

A. The Silent Forest of Congestion Trees

The most basic congestion tree forms when a subset, C, of the nodes, N, in the network constantly injects traffic to a permanent hotspot, while the rest of the nodes, V, injects traffic to other destinations[3]. Then we have $N = C \cup V$, where C are the contributors to congestion, while V are the potential victims suffering from HOL blocking. In such a situation, a stable congestion tree will grow branches from the root of the congestion, ultimately the last link towards the hotspot, along the backward paths towards each of the nodes in C. In an IB network with CC enabled, the branches in such a congestion tree will grow and get pruned as the throttling mechanism constantly tries to adjust the injection rates of the nodes in C to their fair share of the resources at the root of the tree. Notice, however, that the branches will not move. The growing and shortening of a branch always happens along the same path. We refer to a congestion tree with such branches as a *silent* tree, as the tree's branches are *not blowing in the wind*. Figure 2 shows an artificial network topology where a silent congestion tree, represented by the red lines, has grown all the way from the root of the tree to the five source nodes contributing to congestion. We have somewhat artificially gathered all the source nodes in the network at the top, and the destination nodes at the bottom, to make the congestion tree visually more appealing and the characteristics of the tree easier to comprehend.

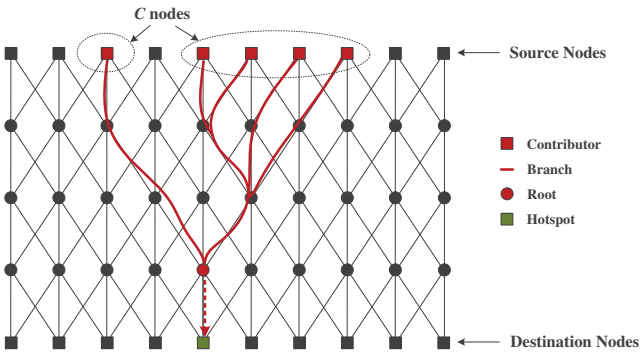


Figure 2. A silent congestion tree.

If C is divided into subsets $C_1, C_2, ..., C_n$ where each subset sends to a different hotspot, the corresponding network will grow a forest of silent congestion trees. Such a forest may resemble a set of nodes sending large virtual image files back to file servers in a network, or a set of sensor nodes continuously transferring large amounts of data back to a set of collector nodes.

The task of the IB CC mechanism will, as always, be to prune the branches of the trees all the way down to their roots, without causing under utilisation of the bottleneck resources. As the number of contributors is constant and the hotspots are stable, the IB CC should have a relatively easy task, given enough time, to adjust the injection rate of each contributor to its fair share of the bottleneck resources. The throttling mechanism at a source node does not need to keep track of different injection rates related to different destinations or traffic flows, and the recovery process after a period of congestion has ended is not really challenged as the hotspots are always present.

[3]Notice that in general, a node can be a contributor to one congestion tree and a victim of another - at the same time - depending on the communication patterns in the network.

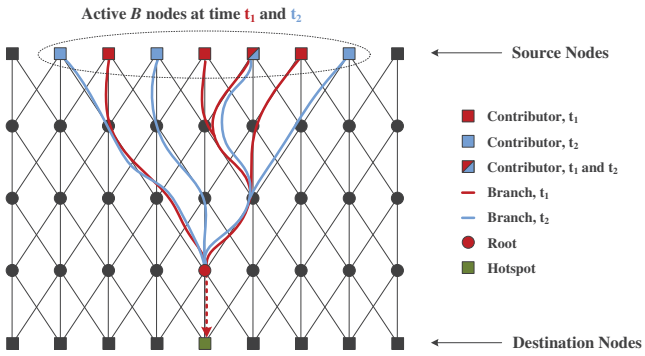


Figure 3. A windy congestion tree.

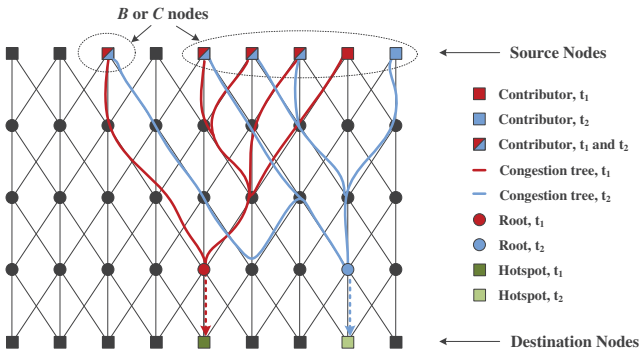


Figure 4. A moving congestion tree.

B. The Windy Forest of Congestion Trees

Now consider a network where the nodes are not divided into pure contributors to and possible victims of congestion, like in the previous section, but instead consists of a different type of nodes, B, that sends a certain percentage p of its traffic to a hotspot, and then the rest $1 - p$ percent to other nodes in the network. A B node can then be both a contributor to and a victim of congestion, depending on the time period we are looking at. In a network of B nodes, the traffic creates a congestion tree, given that p is not too low, but it will be different from a silent one. The root of the congestion tree will as before be permanently related to the hotspot, but the branches will now be more dynamic. No longer will the branches only grow and shrink along a specific set of paths, but the paths themselves change as the nodes actually sending to the hotspot at any given time changes. The branches of the congestion tree will be moving like the branches of a tree blowing in the wind. We have created a *windy* congestion tree. Figure 3 shows a network topology where seven B nodes create such a tree. First, at time t_1, four nodes create a set of branches for the congestion tree, the red lines, while later, at time t_2, a different set of nodes create a different set of branches, the blue lines. The branches have moved, while the root is the same. Notice that even though we in figure 3 show the branches as having grown all the way back to the source nodes, this will most likely not be the case in most situations. The branches will usually have different lengths, depending on the traffic pattern and the value of p.

Again, if B is divided into subsets $B_1, B_2, ..., B_n$ where each subset sends p_i percent to its own hotspot, the corresponding network will grow a forest of windy congestion trees. Such a traffic scenario could bear a resemblance to a set of compute nodes that communicate and exchange data with their peers, while at the same time store data at a set of storage nodes, i.e. the hotspots. The ratio between the peer communication and the storage usage is then given by the value of p.

The windy forest of congestion trees poses a new challenge for the IB CC mechanism. The branches of the congestion trees need to be pruned as before, but now the number of contributors are varying depending on p and what nodes that happen to send to the hotspot at a given time. In addition, it becomes important for the IB CC mechanism to separate the injection rate of different flows, to make sure that traffic not headed for the hotspot is not held back by the throttling mechanism. It becomes important that the IB CC operates at the QP level (or at least at the source-destination-pair level), and not the SL/VL level. The recovery process after a period of congestion has ended is, however, still not really challenged as the hotspots are always the same.

C. The Forest of Moving Congestion Trees

Both the silent and the windy congestion trees have a root related to a permanent hotspot. Now let us assume that the contributors to congestion sending to the hotspot H_i, no matter if they are contributing to a silent or windy congestion tree, change their target destination to a new hotspot, hotspot H_j. Independent of the congestion control mechanism, the congestion tree caused by the traffic headed for H_i will now have its root moved towards a new root location related to the new hotspot H_j, where new branches will grow. The congestion tree has *moved*[4]. Figure 4 illustrates a moving congestion tree before and after a set of contributors have changed focus from one hotspot at time t_1, to another at t_2. The red lines correspond to the congestion tree at t_1, while the blue lines correspond to the congestion tree at t_2. Again, the branches are for clarity shown as having grown all the way back to the source nodes.

A forest of moving congestion trees has a more dynamic nature than a forest of silent or windy congestion trees, no

[4]Strictly speaking, what is happening is that one tree disappears as a new one is created. The original congestion tree is not actually moving as such, but as both events happen at the same time, with contributors changing focus from one hotspot to another, we will think of it as a congestion tree moving from one part of the network to another.

matter if the contributors of the moving trees are C or B nodes, or both. As we shorten the lifetime of each hotspot, the forest of moving congestion trees becomes increasingly more dynamic. Different trees will grow and shrink around the network in a continuously more nonsystematic way, and the traffic pattern as a whole is moving closer to a chaotic scenario where knowledge about the traffic is limited. Such a traffic scenario could resemble a cluster running a set of virtual machines or virtual jobs, where the communication pattern is unknown, depending on the jobs currently running. Short (and long) lived congestion can appear anywhere in the network at any given time, depending on the lifetime of the hotspots, the number of contributors and their nature.

In a forest of moving congestion trees, the IB CC mechanism faces new challenges as the dynamics increases. Not only is it still important to detect congestion fast, decrease the injection rates accordingly, and separate flows contributing to congestion from other flows at a source node, but it also becomes important to recover fast from congestion. That is, the injection rate of a flow contributing to congestion should at a source node be increased again fast enough to ensure good utilization of network resources as soon as the congestion disappears. It is an indisputable fact that a feedback control loop like the one the IB CC mechanism relies upon, can lead to a CC mechanism constantly operating (too far) behind schedule as it takes time to bring information about the congestion from the root of the congestion tree back to the contributors. The shorter the lifetime of the hotspots, the more challenging it will be for the CC mechanism to react fast enough to keep up with the actual situation in the network, both when decreasing the injection rate as congestion is detected, as well as increasing it again when congestion is resolved. An interesting question is then if this really poses a problem, as the shorter lifetime of the hotspots, and the correspondingly more dynamic traffic pattern, by itself implies that the occurrences of congestion will be less severe, and the negative effects of congestion less dramatic. The dynamic traffic pattern will by its own nature reduce the severity of the HOL blocking in the network.

The Diverse and Stormy Forest of Moving Congestion Trees

It can be argued that the classification of congestion trees given above is somewhat artificial. In a network, even if the main contributors to congestion create a silent congestion tree, other small and short lived congestion trees will be created by other traffic flows present in the network. In addition, HOL blocking will make congestion trees grow towards victims of congestion, and not only towards the initial contributors like shown in figure 2. Furthermore, a silent congestion tree can be seen as a windy congestion tree with $p = 1$, and a moving congestion tree can be seen as two separate events where one congestion tree disappears while another one is created. Typically, in a network, we will find a multitude of different congestion trees that all have their own twists. The congestion trees truly come in all kinds of flavors.

Still, even if a network in general can grow a diverse and stormy forest of moving congestion trees, the classification of congestion trees into silent, windy, and moving trees has proved to be very useful when conducting a systematic study of the IB CC mechanism's performance as the degree of dynamic behaviour in the network increases. By controlling the type of main contributors to congestion, being C or B nodes, and the duration of each hotspot, we control the main types of congestion trees that will form in the network. The main congestion trees are the ones causing the most HOL blocking, and by that the main reasons for the performance degradation observed in the network. Then, by starting with contributors that create silent trees, and later moving on to nodes creating windy and moving congestion trees, we can study how well the IB CC mechanism is able to cope with an increasingly more dynamic traffic pattern in the network. In addition, notice that even though we focus on the three different types of congestion trees, in our simulations (as in a real network) any background traffic and the corresponding HOL blocking introduced, will obviously both create a multitude of small congestion trees, as well as make the main trees grow towards victim nodes. This behavior will be captured and included in the overall network performance measurements presented in section V.

IV. The Simulator

Our network simulator and switch model is built on the OMNet++ platform [19] and has been previously described in [20]. Below we give a brief overview of the model and the simulation parameters used in all the simulations presented in section V. The IB model consists of a set of simple and compound 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) compound module and the *Switch* compound module. The *Switch* consists of a set of *SwitchPort*s, which are by themselves compound modules. During a simulation, an HCA represents both a traffic injector and a traffic sink in the network, while a *Switch* acts as a forwarding node. The HCAs and *switches* are connected using *gates*, corresponding to links in the network.

The HCAs and the *SwitchPort*s consist of the a set of common simple modules *ibuf*, *obuf*, *vlarb*, and *ccmgr*, while the simple modules *gen* and *sink* are exclusive to the HCAs. The *ibuf* represents an input buffer with support for virtual lanes, virtual output queuing (VoQ) and virtual cut through switching. The *obuf* represents a simple output buffer, while the *vlarb* implements round robin arbitration over the

different VLs and multiple input ports (if the module is part of a *SwitchPort*). The *gen* implements traffic generation in a HCA, while the *sink* is the part of the HCA responsible for removing traffic from the network. The *gen* module supports several traffic generation schemes, e.g. varying the injection rate, the packet size and the destination node distribution.

In general, the *gen* module at an HCA generates traffic that is forwarded through the *vlarb* to the *obuf* of the HCA. From there it is sent out into the network. There is no internal HoL blocking in the *gen* module or in the HCA so the only place where the traffic can experience blocking is in the switches. At a *Switch*(*Port*) the *ibuf* receives the traffic, does the routing decision and moves the traffic into the corresponding VoQ. Here the traffic waits until the *vlarb* of the given output port grants access to the corresponding *obuf*. At an HCA the *ibuf* receives traffic and forwards it to the *sink*. The IB flow control is managed by the *ibuf*s and the *obuf*s, exchanging flow control messages.

The InfiniBand Congestion Control mechanism is implemented by the simple module *ccmgr*. The *ccmgr* is included in the compound modules for the HCA and the *SwitchPort*, and manages everything related to congestion control in these modules, with the help of the other simple modules therein. In particular, all CC parameters specified in the InfiniBand Architecture Specification release 1.2.1 [9] are supported by the *ccmgr* module. The throttling mechanism operates at the packet level. That is, all the flits belonging to the same packet are always injected back-to-back, if allowed by the link-level flow control mechanism, even when the throttling mechanism is active. The IRD calculations follow the IB specification. The simulation model, including the CC behaviour, has been carefully tuned against Mellanox MTS3600 InfiniBand switches as described in [20].

As our simulation scenario we have selected a three stage fat-tree network, which is a topology common in large scale InfiniBand switches [21], [22]. This topology supports non-blocking communication of up to 648 compute nodes and is built from 54 36-port crossbars. In our simulations we used a link speed of 20 Gbit/s (4x DDR) and an MTU size of 2048 bytes. Each message sent by a node consists of two packets, giving a total size of 4096 bytes per message. A node injects packets at full link speed whenever possible. The traffic patterns used correspond to the scenarios described in section III, while the specific destination distributions used are given during the discussion of the results in section V. Frame I gives a detailed example of how traffic is generated at a node during a simulation, here using a B node with $p = 50$.

The congestion control parameter values used during all the simulations discussed in this paper are listed in table I. These values were found during our initial study of CC capable IB hardware, a work presented in [7]. While it proved to be a nontrivial task to identify the parameter values, even in a small cluster with a simple and static traffic pattern, the values found in [7] have proved to be quite robust, as we will see in section V. Note that the CCT values have been increased to reflect the larger number of possible contributors to congestion in our fat-tree topology, compared to our earlier hardware experiments.

Frame I - *packet generation at a* B *node with* p=50:

A B node with $p = 50$ sends 50% of its generated traffic to its designated hotspot, hs, while the rest is sent using a uniform destination distribution including all nodes in the network (expect the node itself). Using a message size of two packets, corresponding to a total of 4096 bytes per message, the sequence of packets generated and sent could then look like this:

- 2 x Msg sent with random destination addresses (most likely two different destinations)
- 1 x Msg sent to hotspot hs (=4KB)
- 1 x Msg sent with random destination address
- 3 x Msg sent to hotspot hs (=12KB)
- 2 x Msg sent with random destination addresses (most likely two different destinations)
- 1 x Msg sent to hotspot hs (=4KB)
- ... and so on.

In a scenario of moving congestion trees, the B node changes the address of the hotspot at each new timeslot (e.g. each 1msec); the hs_i address changes into hs_j, and by that the hotspot is moved, while the value of p remains the same.

The packets are sent back-to-back when not held back by the CC mechanism or the link-level flow control. Note though, that the $p\%$ and $(1-p)\%$ fractions of a B node are related to the (simulation) time, and not each other. That is, after a given time, t, a maximum of $p\%$ of the traffic has been sent to the hotspot, while a maximum of $(1-p)\%$ has been sent to the non-hotspots. The link will remain idle when non-hotspot traffic has fulfilled its $(1-p)\%$ of the total possible traffic sent (t times link capacity) and the hotspot traffic is held back by the CC mechanism. It is important to keep the two types of traffic (hotspot and non-hotspot) independent of each other, i.e. it is important that non-hotspots traffic is not HOL blocked internally in the generator when the hotspot traffic is held back by the CC mechanism, while at the same time it is just as important that the non-hotspot traffic does not exceed the $(1-p)\%$ during a simulation as this is the amount of traffic supposedly requested by non-hotspot traffic during a simulation time.

Note that by changing the value of p, the fraction of traffic going to the hotspot changes, and by that, the likely (and average) lengths of the *trains of packets* going to the hotspot changes as well. When using a B node with a p value of 50%, most trains going to the hotspot will have sizes in the range of [4KB, a few KB> while most trains going to non-hotspots will have the size of a single message, 4KB, due to the uniform destination distribution.

Table I
CC PARAMETER VALUES.

Parameter	Value	Parameter	Value
$CCTI\_Increase$	1	$Threshold$	15
$CCTI\_Limit$	127	$Marking\_Rate$	0
$CCTI\_Min$	0	$Packet\_Size$	0
$CCTI\_Timer$	150		

V. SIMULATION RESULTS

We have divided the presentation and the analysis of our simulation results into three sections, corresponding to the

Table II
PERFORMANCE NUMBERS (GBPS), SILENT CONGESTION TREES.

No hotspots, no CC	
Avg. receive rate	2.699
No hotspots, CC on	
Avg. receive rate	2.701
Hotspots, no CC	
Hotspots avg. rcv.	13.602
Non-hotspots avg. rcv	0.168
Hotspots, CC on	
Hotspots avg. rcv.	13.279
Non-hotspots avg. rcv	2.246
Totale Network Throughput, Hotspots	
Without CC	216.073
With CC	1543.793

three categories of congestion trees introduced in section III. In the first section, section V-A, we start by looking at the performance of the IB CC mechanism in a network growing a quiet forest of eight silent congestion trees. Then, in section V-B, we continue our study by gradually exchanging the nodes in the network with B nodes, making the congestion trees increasingly more windy, and by that, the traffic pattern in the network more dynamic. Finally, in section V-C, we end our study by looking at the performance of IB CC as the hotspots move. By increasing the number of times the hotspots move during a given timeslot, we decrease the hotspot lifetimes accordingly, and by that, the dynamics of the traffic pattern in the network increases even further.

A. The Silent Forest of Congestion Trees

In this scenario, the end nodes in the network are divided into 80% C nodes and 20% V nodes. Recall that this means that 80% of the 648 nodes in the network send traffic solely to the hotspots. The rest of the nodes, the V nodes, send traffic with a uniform destination distribution. The V nodes are randomly distributed in the topology. All nodes constantly try to inject traffic into the network at maximum capacity; 13.5Gbps[5].

Before enabling the C nodes, we simulate a scenario where only the V nodes are active and the CC mechanism disabled. Doing this, we get the throughput of the nodes in the network that potentially will be victims of congestion when the hotspots are introduced. As shown in the first part of table II, the average receive rate of the nodes in the network is approximately 2.7Gbps. This is as expected, as each V node injects traffic at 13.5Gbps into the non-blocking topology. The soon-to-be hotspots and the non-hotspots receive the same amount of traffic. These performance numbers remain the same if we enable CC, still without activating the C nodes (the second part of table II). Enabling CC causes no harm to the network performance in this lightly loaded network.

Now let us disable the CC again, and enable the C nodes. The C nodes are evenly divided into eight subsets, each subset sending to one of eight hotspots. The third part of table II shows the average receive rates of both the newly created hotspots and the non-hotspots. The hotspots, randomly distributed in the network, each receives 13.6Gpbs[6], while the non-hotspots have their average receive rate lowered from 2.7Gpbs down to 0.168Gbps. The growth of the eight silent congestion trees results in a huge amount of HOL blocking in the network, and by that a severe performance degradation for the V nodes. The V nodes have become victims of congestion. To remove the congestion trees and the HOL blocking, we turn the CC back on. The fourth part of table II shows the resulting improvement in performance. At the cost of a small drop in the receive rate for the hotspots, down 2.5%, we improve the receive rate of the non-hotspots by more than 1200%. Enabling the CC mechanism of IB, the non-hotspots now experience a receive rate only 17% below the receive rate they achieved before the hotspots were introduced. The last part of table II shows the total network throughput of the network running with and without CC enabled. Even when accounting for the drop in the receive rate of the hotspots when CC is enabled, enabling CC leads to a performance improvement by more than 610%. The CC mechanism, using the CC parameters from Table I, is clearly able to improve the performance in our network when silent congestion trees are present.

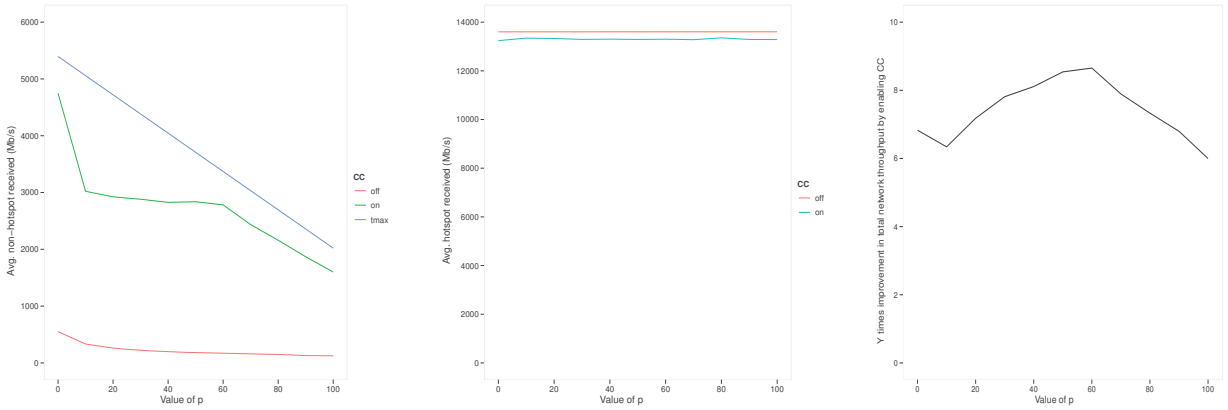
B. The Windy Forest of Congestion Trees

Let us now increase the dynamics of the traffic pattern used in the previous section by gradually exchanging $x\%$ of the nodes in the network with B nodes, increasing x in steps of 25%. The eight permanent hotspots are still present, but as x increases so does the amount of wind in our congestion trees. The B nodes are evenly divided into eight subsets, just like the C nodes, each subset sending to one of the eight hotspots. The nodes not being B nodes, i.e. $(100 - x)\%$ of the nodes, are divided into 80% C nodes and 20% V nodes – as before. Note that until x reaches 100%, this implies that there are always some permanent contributors to congestion present in the network (and some permanent potential victims). The figures 5, 6, 7, and 8 show performance plots from simulations ran with $x = 25\%$, $x = 50\%$, $x = 75\%$, and $x = 100\%$, respectively. These four scenarios and their corresponding figures will be discussed in the following subsections.

1) 25% B Nodes: The figures 5(a) and 5(b) show the average receive rates of non-hotspots and hotspots, respectively, in a network running with and without CC, as p

[5]This corresponds to the injection rate of the end nodes the simulator is tuned against, an injection rate limited by the PCIe v1.1 protocol overhead and other system components in the hardware.

[6]This matches the receive rate of the end nodes the simulator is tuned against. The hardware has a receive rate approximately 0.1Gbps higher than the injection rate.



(a) Average receive rate, non-hotspots. (b) Average receive rate, hotspots. (c) Total network throughput improvement.

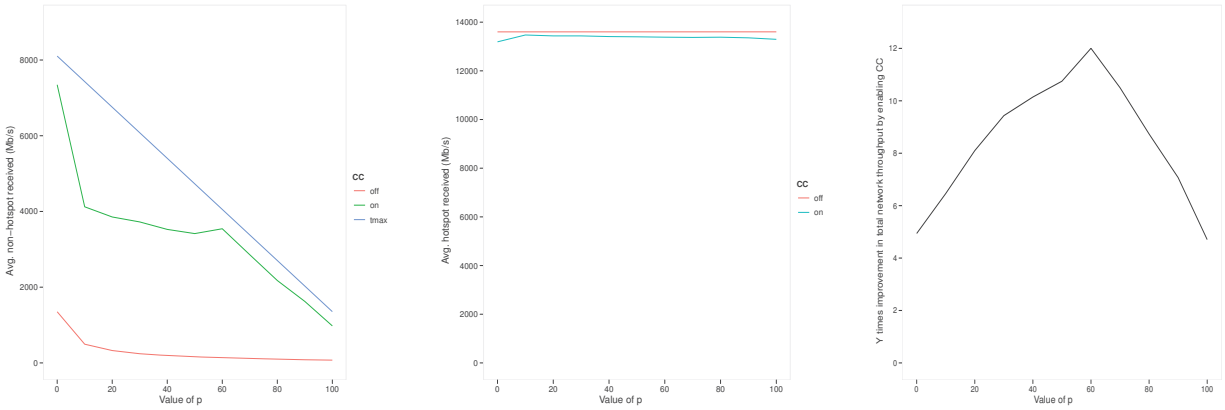
Figure 5. Measurements for a windy forest with 25% B nodes as p increases.

increases from 0 to 100. Recall that a B node sends p% of its traffic to a given hotspot. The remaining $(1-p)$% of the traffic is sent using a uniform destination distribution. Note that this implies that as p increases, a decreasing amount of traffic in the network is destined for the non-hotspots. That is, as p increases, the theoretical maximum amount of traffic that the non-hotspots can receive decreases. This theoretical maximum is plotted as *tmax* in figure 5(a), and represents the maximum average receive rate the non-hotspots could possibly achieve if the hotspots were not present.

Looking at figure 5(a), it is evident that enabling CC in the network results in an immense improvement in the average receive rates for the non-hotspots, independent of the value of p. At $p = 0$ the average receive rate of a non-hotspot is 4.75Gbps when CC is enabled, compared to 0,55Gbps when CC is disabled and the tmax value of 5.4Gbps. In this case, enabling CC leads to an 8.6 times improvement in performance; a 760% increase. As p increases from 0 to 10, the performance when CC is enabled drops from 88% to 60% of tmax, while the relative performance actually increases to a factor of 9.1 compared to the scenario with CC disabled. As the B nodes start to send traffic to the hotspots, but still generate 90% uniformly distributed traffic, the CC mechanism is not fully able to cope with all the HOL blocking in the network. As p increases from 10 to 60, however, the relative performance of CC compared to tmax increases to 80% again, where it stays as p reaches the value of 100. The performance increase by enabling CC in the network, as the p values increases from 30 to 100, equals to a factor ranging from 12.9 to 16.3, with the peak at $p = 60$; a 1530% performance boost in the receive rate of the non-hotspots at this point.

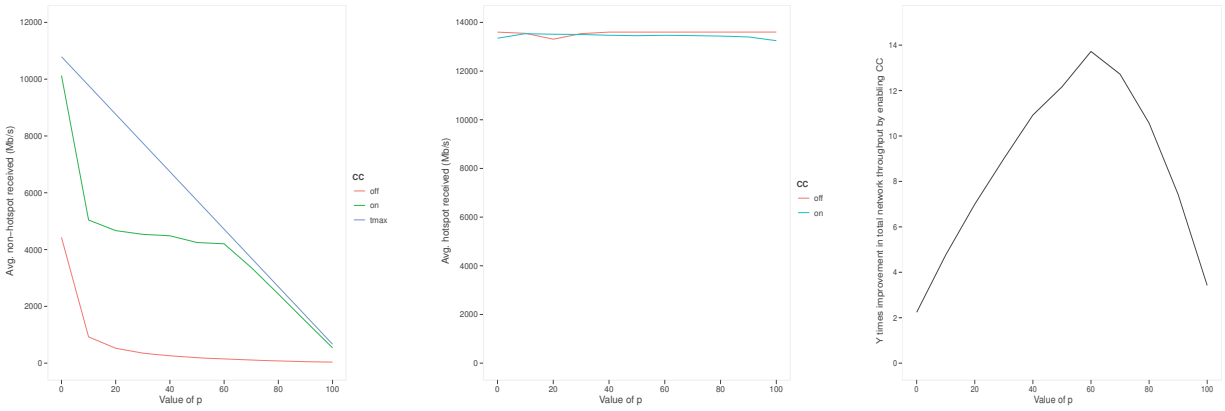
Turning our attention towards the hotspots, figure 5(b) shows that the average receive rates of these nodes are independent of the values of p. When CC is disabled, each hotspot receives a steady 13.6Gbps, which equals to the maximum receive rate of an end node. The average receive rate then drops by 2.2%, down to 13.3Gbps, when the CC is enabled. While this indicates a tiny underutilization of the scarce resources at the roots of the congestion trees, bearing the results from figure 5(a) in mind, the price we have to pay when enabling CC is negligible. Figure 5(c) plots the improvement in the total network throughput when CC is enabled as a function of p. By enabling CC, we improve the total network throughput by a factor ranging from 6.0 ($p = 100$) to 8.7 ($p = 60$). That is, a minimum improvement in performance by at least 500%.

2) 50% and 75% B Nodes: Figures 6 and 7 show performance plots from simulations where the fraction of B nodes in our network is 50% and 75%, respectively. Comparing these results with the ones presented in figure 5, we observe that the performance trends are the same. Enabling CC leads to a vast improvement in the average receive rates of the non-hotspots, without penalizing the average receive rates of the hotspots. Notice that as the fraction of B nodes increases and the fraction of C and V nodes decreases, the impact the value of p has on the overall traffic pattern in the network increases accordingly. This influences the tmax values of the average receive rates of the non-hotspots. At $p = 0$, the tmax value increases as the fraction of B increases, because the traffic pattern in the network as a whole then moves towards a uniform destination distribution. On the other hand, at $p = 100$, the tmax value decreases as the fraction of B nodes increases. At this p value, a B node sends all its traffic to a hotspot, and then as the fraction of B nodes increases, the fraction of the traffic in the network headed for the hotspots increases accordingly. To sum up, the graph of decreasing tmax values as a function of p, becomes steeper as the fraction of B nodes in the network increases. This influences the possible



(a) Average receive rate, non-hotspots. (b) Average receive rate, hotspots. (c) Total network throughput improvement.

Figure 6. Measurements for a windy forest with 50% B nodes as p increases.

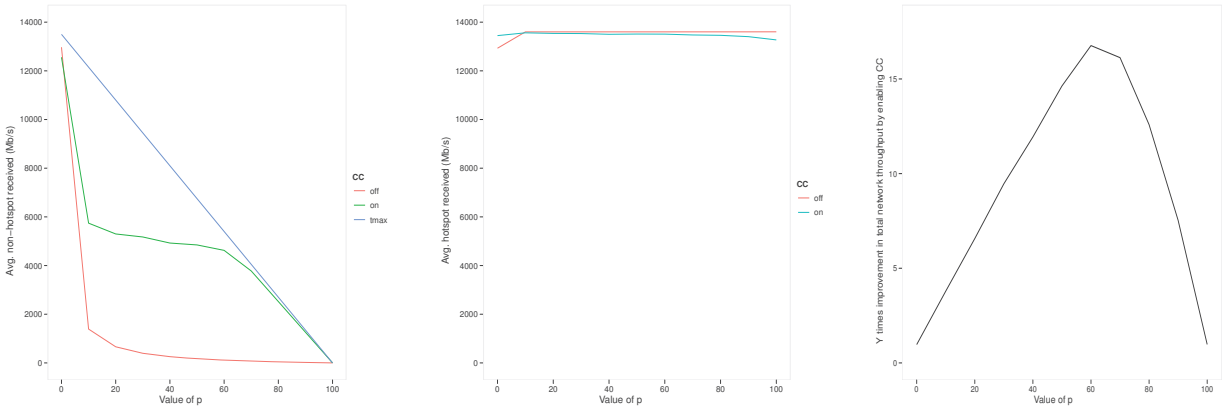


(a) Average receive rate, non-hotspots. (b) Average receive rate, hotspots. (c) Total network throughput improvement.

Figure 7. Measurements for a windy forest with 75% B nodes as p increases.

benefit we can achieve by enabling CC in our network, an effect clearly present in the graphs showing the total network throughput improvement (figure 6(c) and 7(c)). The graphs become increasingly more $\cap$ shaped as the fraction of B nodes increases. At low and high p values the total network throughput improvement decreases, while the peak at $p = 60$ increases. This effect becomes even clearer as we increase the fraction of B nodes to 100%.

3) 100% B Nodes: Figure 8 shows the performance plots as the fraction of B nodes has increased to 100%. We are creating purely windy congestion trees with no traffic generated by V and C nodes. Now, for the first time, we observe a small penalty of 3% in the receive rate of the non-hotspots at $p = 0$ as CC is enabled (figure 8(a)). Note though that at this p value, the B nodes have no preference for the hotspots. The traffic is uniformly distributed in the network, and there is no real congestion for the CC to resolve. As soon as the B nodes start to send traffic to the hotspots (at $p > 0$), the improvement by enabling CC is again evident. At $p = 10$, enabling CC leads to a 4.1 times improvement in the receive rate of the non-hotspots, an improvement increasing to 64.1 times as p approves 90. In this scenario, the non-hotspots experience a total collapse in performance in a network without CC, while when enabling CC, the same nodes experience a receive rate very close to the theoretical maximum when $p > 60$. Figure 8(b) shows that enabling CC has no negative effect on the receive rate of the hotspots. Finally, looking at the total network throughput improvement in figure 8(c), the IB CC mechanism's ability to improve network performance when congestion is present in the network is clear. At $p = 0$ and $p = 100$, enabling CC has virtually no effect, as the CC mechanism is left with no room for improvement - as explained in the previous section. Note though, that the CC mechanism does not



(a) Average receive rate, non-hotspots. (b) Average receive rate, hotspots. (c) Total network throughput improvement.

Figure 8. Measurements for a windy forest with 100% B nodes as p increases.

cause any harm at these extreme p values. With p values in the interval $< 0, 100 >$, the CC mechanism shows a performance improvement, with a peak at p=60, leading to a seventeen-fold increase in total network performance.

Summing up, a network with CC enabled outperforms a network running without CC as long as congestion is present in the network, no matter how windy the congestion trees are, showing a peak in performance when approximately 60% of the traffic is headed towards the hotspots. Furthermore, the CC mechanism causes no harm to the network performance when all traffic is headed for the hotspots, and the reduced performance caused by CC in a network with purely uniformly distributed traffic is negligible.

C. The Stormy Forest of Moving Congestion Trees

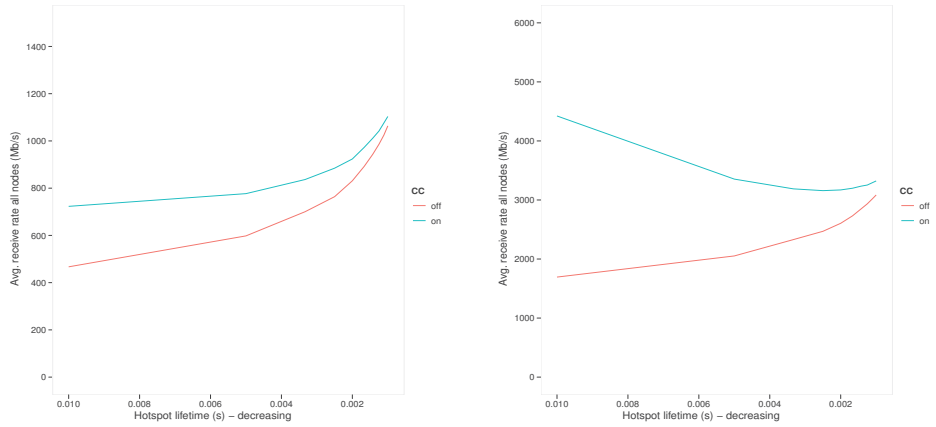
This far, all our hotspots were locked to a permanent position in the network during a simulation. To continue our study of the performance of the IB CC mechanism as the dynamics in the network increases, we now start to move the hotspots. During a simulated timeslot of 0.1s, the hotspots are moved n times, n ranging from 10 to 100. This corresponds to a shortening of the hotspot lifetimes from 10ms to 1ms. By measuring the performance of the IB CC for different values of n, we can to study the performance of the CC mechanism in a systematic way as the dynamics in the network increases. The CC now needs to deal with contributors that themselves, dynamically, tear down and recreate congestion trees in the network. In addition, the moving process itself may create temporarily congestion trees at unforeseeable places in the network. By moving the hotspots, we turn our network into a stormy forest of moving congestion trees.

We start our study by moving silent congestion trees, before we move on to a scenario where we move windy congestion trees. The contributors are as before divided into eight subsets, each subset sending to one of the eight hotspots.

Figure 9(a) shows the average receive rate of all nodes in a network consisting of 20% V nodes and 80% C as a function of the decreasing hotspot lifetime. When the hotspots are moved each 10th ms, the nodes receive 723Mbps when CC is enabled, compared to 467Mbps in a network without CC. The performance increases by 55% when enabling CC. Then, as the lifetime of the hotspots are shortened, the improvement in performance by enabling CC is reduced. When the hotspots move every second millisecond, the performance increase is down to 10%, and as the hotspot lifetime reaches 1ms, the performance increase is only 4%. The performance increase is less impressive than the ones achieved when the hotspots are not moving, but the benefit from enabling the IB CC mechanism is still clear even as the hotspot lifetime approaches 1ms.

Note that as the lifetime of the hotspots decreases, the receive rate increases in general, while the advantage from enabling CC decreases. When the hotspot lifetime decreases, the contributors change focus more often and by that they actually distribute the traffic more evenly in the network. The result is that the network throughput as a whole increases, as it is less dependent on the limited number of hotspots' ability to receive traffic. At the same time, the change in focus of the contributors will help to resolve congestion in the network. This also implies that the CC mechanism is left with less room for improvement. In addition, when the lifetime of the hotspots decreases, it becomes increasingly hard for IB CC mechanism to keep up with the situation in the network (due to the feedback loop).

A network with 20% V nodes and 80% C nodes has relatively few possible victims of congestion; only 20% of the nodes. Figure 9(b) shows the performance numbers we get if we increased the number of V nodes to 60%. Now,



(a) 20% V nodes, 80% C nodes.

(b) 60% V nodes, 40% C nodes.

Figure 9. Average receive rates in a network with silent congestion trees and moving hotspots, shown as a function of decreasing hotspot lifetimes.

the advantage of enabling CC has increased again when the hotspot lifetime is 10ms. At this point, enabling CC results in an increase in the average receive rate by a factor 2.6. The advantage by enabling CC decreases faster, however, than when having 20% V nodes. At a hotspot lifetime of 1ms, the improvement is down to 10% again.

Finally, figure 10 shows the average receive rate of the nodes in a network consisting of only B nodes, using p values of 30, 60, and 90. As we move the hotspots and their corresponding windy congestion trees, we observe the same performance trends as we did when moving the silent congestion trees. The enabling of CC leads to an improvement in performance in all cases, even though the improvement decreases as the hotspot life time decreases and the traffic pattern itself alleviate the side effects of congestion in the network.

VI. Conclusions

A central question in the understanding of congestion control in interconnection networks is whether throttling of contributors to congestion may have adverse effects. Setting parameters related to the feedback-loop of such mechanisms have previously been shown to require deep understanding of the problem, even for simple traffic. Therefore uncertainty has lingered as to whether congestion control may actually be harmful in some scenarios.

In this paper we have shown that for fat-trees, there exist parameter settings that makes the throttle-based Congestion Control of InfiniBand robust. Through carefully designed experiments we have stressed the mechanism with congestion scenarios varying from the completely static ones, to cases where congestion is highly dynamic both with respect to contributors, intensity, duration and placement of congestion points. Our identified parameter settings showed increased throughput varying from a few percent, to a seventeen-fold increase. The only adverse effect we registered was a negligible decrease in throughput for the contributors to congestion.

Our most important result is that InfiniBand congestion control can be tuned to be stable for a given installation based on fat-trees. The tuning itself remains a highly specialized task [7], but the gains in performance is huge when it is done correctly. Regarding other topologies, the question is still open. There is reason to believe that other multistage-topologies that have a similar pattern of inter-relations between streams, will expose the same behavior. Regarding Tori or Meshes, the picture is more unclear, thus this question should form the basis for further research.

References

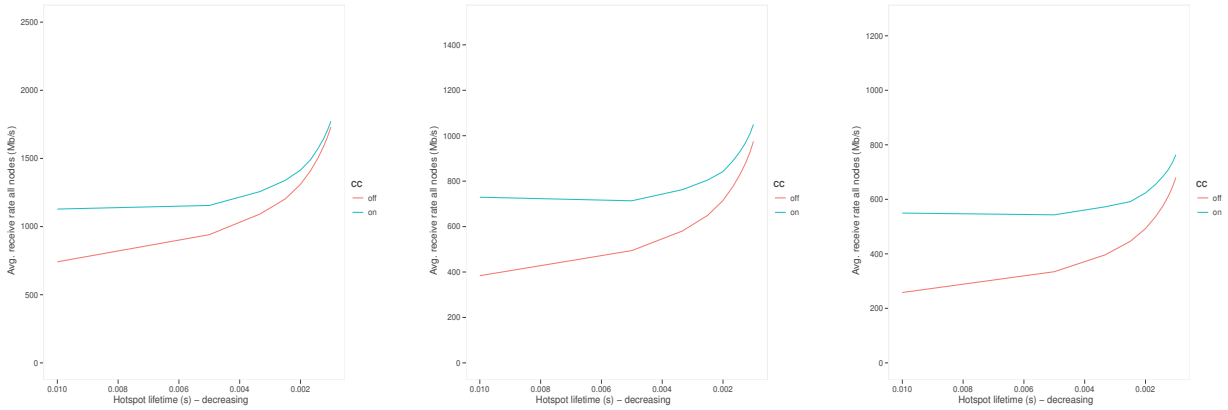
[1] G. F. Pfister and V. A. Norton, ""Hot Spot" contention and combining in multistage interconnection networks," *IEEE Trans. Computers*, vol. 34, no. 10, pp. 943–948, 1985.

[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.



(a) B nodes with $p = 30$. (b) B nodes with $p = 60$. (c) B nodes with $p = 90$.

Figure 10. Average receive rates in a network with 100% B nodes and moving hotspots, shown as a function of decreasing hotspot lifetimes.

[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*, Apr. 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. GarcÃa, 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, P. Sadayappan, M. Parashar, R. Badrinath, and V. Prasanna, Eds. Springer Berlin / Heidelberg, 2008, vol. 5374, pp. 503–517.

[16] "Top 500 supercomputer sites," http://top500.org/, November 2011.

[17] E. G. Gran, E. Zahavi, S.-A. Reinemo, T. Skeie, G. Shainer, and O. Lysne, "On the relation between congestion control, switch arbitration and fairness," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, ser. CCGRID '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 342–351. [Online]. Available: http://dx.doi.org/10.1109/CCGrid.2011.67

[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] "Omnet++ network simulation framework," http://www.omnetpp.org/.

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

[21] Sun Microsystems, "SUN DATACENTER INFINIBAND SWITCH 648 ARCHITECTURE AND DEPLOYMENT," Sun Microsystems, Tech. Rep. June, 2009.

[22] "IS5600 - 648-port InfiniBand Chassis Switch," Mellanox Technologies, http://www.mellanox.com/related-docs/prod_ib_switch_systems/IS5600.pdf.