Making the Network Scalable: Inter-subnet Routing in InfiniBand

Bartosz Bogdański[1], Bjørn Dag Johnsen[1],
Sven-Arne Reinemo[2], and José Flich[3]

[1] Oracle Corporation, Oslo, Norway
{bartosz.bogdanski,bjorn-dag.johnsen}@oracle.com
[2] Simula Research Laboratory, Lysaker, Norway
svenar@simula.no
[3] Universidad Politécnica de Valencia, Valencia, Spain
jflich@disca.upv.es

Abstract. As InfiniBand clusters grow in size and complexity, the need arises to segment the network into manageable sections. Up until now, InfiniBand routers have not been used extensively and little research has been done to accommodate them. However, the limits imposed on local addressing space, inability to logically segment fabrics, long reconfiguration times for large fabrics in case of faults, and, finally, performance issues when interconnecting large clusters, have rekindled the industry's interest into IB-IB routers. In this paper, we examine the routing problems that exist in the current implementation of OpenSM and we introduce two new routing algorithms for inter-subnet IB routing. We evaluate the performance of our routing algorithms against the current solution and we show an improvement of up to 100 times that of OpenSM.

1 Introduction

Until recently, the need for routers in InfiniBand (IB) networks was not evident and all the essential routing and forwarding functions were performed by layer-2 switches. However, with the increased complexity of the clusters, the need for routers becomes more obvious, and leads to more discussion about native IB routing [1,2,3]. Obsidian Research was the first company to see the need for routing between multiple subnets, and provided the first hardware to do that in 2006 [4].

There are several reasons for using routing between IB subnets with the two main being address space scalability and fabric management containment. Address space scalability is an issue for large installations whose size is limited by the number of available *local identifiers* (LIDs). Hosts and switches within a subnet are addressed using LIDs and a single subnet is limited to 49151 unicast LIDs. If more end-ports are required, then the only option is to combine multiple subnets by using one or more IB routers. Because LID addresses have local visibility, they can be reused in the subnets connected by routers, which theoretically yields an unlimited addressing space. It is worth observing that there are multiple suggestions to expand the address space of IB without introducing

F. Wolf, B. Mohr, and D. an Mey (Eds.): Euro-Par 2013, LNCS 8097, pp. 685–698, 2013.

routers. One of the more mature proposals aims at extending the LID addressing space to 32 bits [5], however, it would not be backward compatible with older hardware, which limits its usability.

Fabric management containment has three major benefits: 1) fault isolation, 2) increased security, and 3) intra-subnet routing flexibility. First, by dividing a large subnet into several smaller ones, faults or topology changes are contained to a single subnet and the subnet reconfiguration will not pass through a router to other subnets. This shortens the reconfiguration time and limits the impact of a fault. Second, from a security point of view, segmenting a large fabric into subnets using routers means that the scope of most attacks is limited to the attacked subnet [6]. Third, from a routing point of view, fabric management containment leads to more flexible routing schemes. This is particularly advantageous in case of a hybrid fabric that consists of two or more regular topologies. For example, a network may consist of a fat-tree part interconnected with a mesh or a torus part (or any other regular topology). The problem with managing this in a single subnet is that it is not straightforward to route each part of the subnet separately because intra-subnet routing algorithms have a subnet scope. Moreover, there are no general purpose agnostic routing algorithms for IB that will provide optimal performance for a hybrid topology. However, if a hybrid topology is divided into smaller regular subnets then each subnet can be routed using a different routing algorithm that is optimized for a particular subnet. For example, a fat-tree routing algorithm could route the fat-tree part and the dimension-order routing could route the mesh part of the topology. This is because each subnet can run its own subnet manager (SM) that configures only the ports on the local subnet and routers are non-transparent to the subnet manager.

In this paper, we present two inter-subnet routing algorithms for IB. The first one, *inter-subnet source routing* (ISSR), is an agnostic algorithm for interconnecting any type of topology. The second one is fat-tree specific and only interconnects two or more fat-trees. With these algorithms we solve two problems: how to optimally choose a local router port for a remote destination and how to best route from the router to the destination. We compare the algorithms against the solution that is implemented in OpenSM. Inter-subnet routing in OpenSM is at the time of writing very limited, the configuration is tedious and the performance is only usable for achieving connectivity - not for high performance communication between multiple sources and destinations [2]. It is, however, the only available inter-subnet routing method for IB.

The rest of this paper is organized as follows: we discuss related work in Sect. 2, and we introduce the IB Architecture in Sect. 3. We follow with a description of our proposed layer-3 routing for IB in Sect. 4. Next, we continue with a presentation and discussion of our results in Sect. 5. Finally, we conclude in Sect. 6.

2 Related Work

Obsidian Strategics was the first company to demonstrate a device marketed as an IB-IB router (the Longbow XR) in 2006 [4]. That system highlighted the need

for subnet isolation through native IB-IB routing. The Longbow XR featured a content-addressable memory for fast address resolution and supported up to 64k routes. The drawbacks of the router included a single 4x SDR link, and its primary application was disaster recovery - it was aimed at interconnecting IB subnets spanning large distances as a range extender. Furthermore, the Longbow XR appears to the subnet manager as a transparent switch, so the interconnected subnets are merged together into one large subnet. When releasing the router, Obsidian argued that while the IB specification 1.0.a defines the router hardware well, the details of subnet management interaction (like routing) are not fully addressed. This argument is still valid for the current release of the specification [7]. In 2007, Prescott and Taylor verified how range extension in IB works for campus area and wide area networks [8]. They demonstrated that it is possible to achieve high performance when using routers to build IB wide area networks. However, they did not mention the deadlock issues that can occur when merging subnets, and they only focused on remote traffic even though local traffic can be negatively affected by suboptimal routing in such a hybrid fabric. In 2008, Southwell presented how native IB-IB routers could be used in System Area Networks [1]. He argued that IB could evolve from being an HPC-oriented technology into a strong candidate for future distributed data center applications or campus area grids. While the need for native IB-IB routing was well-demonstrated, Southwell did not address the routing, addressing and deadlock issues. In 2011, Richling et al. [9] addressed the operational and management issues when interconnecting two clusters over a distance of 28 kilometers. They described the setup of hardware and networking components, and the encountered integration problems. However, they focus on IB-IB routing in the context of range extension and not on inter-subnet routing between local subnets.

When reviewing the literature, we noticed that the studies of native IB-IB routing is focused on disaster recovery and interconnection of wide area IB networks. Our work explores the foundations of native IB-IB routing in the context of performance and features in inter-subnet routing between local subnets. Furthermore, we assume full compliance with the IB specification and we deal with issues previously not mentioned including the deadlock problem and path distribution.

3 The InfiniBand Architecture

InfiniBand is a serial point-to-point full-duplex interconnection network technology, and was first standardized in October 2000 [7]. The current trend is that IB is replacing proprietary or low-performance solutions in the high performance computing domain [10], where high bandwidth and low latency are the key requirements. The de facto system software for IB is OFED developed by dedicated professionals and maintained by the OpenFabrics Alliance [5].

Every IB subnet requires at least one subnet manager (SM), which is responsible for initializing and bringing up the network, including the configuration of all the IB ports residing on switches, routers, and host channel adapters (HCAs) in the subnet. At the time of initialization the SM starts in the *discovering state*

where it does a sweep of the network in order to discover all switches and hosts. During this phase it will also discover any other SMs present and negotiate who should be the master SM. When this phase is completed the SM enters the *master state*. In this state, it proceeds with LID assignment, switch configuration, routing table calculations and deployment, and port configuration. At this point the subnet is up and ready for use. After the subnet has been configured, the SM is responsible for monitoring the network for changes.

A major part of the SM's responsibility is routing table calculations. Routing of the network aims at obtaining full connectivity, deadlock freedom, and proper load balancing between all source and destination pairs in the local subnet. Routing tables must be calculated at network initialization time and this process must be repeated whenever the topology changes in order to update the routing tables and ensure optimal performance. Despite being specific about intra-subnet routing, the IB specification does not say much about inter-subnet routing and leaves the details of the implementation to the vendors.

IB is a lossless networking technology, and under certain conditions it may be prone to *deadlocks* [11,12]. Deadlocks occur because network resources such as buffers or channels are shared and because packet drops are usually not allowed in lossless networks. The IB specification explicitly forbids IB-IB routers to cause a deadlock in the fabric irrespective of the congestion policy associated with the inter-subnet routing function. Designing a generalized deadlock-free inter-subnet routing algorithm where the local subnets are arbitrary topologies is challenging. In this paper we limit our scope to fat-tree topologies and by making sure our routing functions use only the standard up/down routing mechanism, we eliminate the deadlock problem.

3.1 Native InfiniBand Routers

The InfiniBand Architecture (IBA) supports a two-layer topological division. At the lower layer, IB networks are referred to as subnets, where a subnet consists of a set of hosts interconnected using switches and point-to-point links. At the higher level, an IB fabric constitutes one or more subnets, which are interconnected using routers. Hosts and switches within a subnet are addressed using LIDs and a single subnet is limited to 49151 LIDs. LIDs are local addresses valid only within a subnet, but each IB device also has a 64-bit *global unique identifier* (GUID) burned into its non-volatile memory. A GUID is used to form a GID - an IB layer-3 address. A GID is created by concatenating a 64-bit subnet ID with the 64-bit GUID to form an IPv6-like 128-bit address. In this paper, we when using the term GUID we mean a port GUIDs, i.e. the GUIDs assigned to every port in the IB fabric.

IB-IB routers operate at the layer-3 of IB addressing hierarchy and their function is to interconnect layer-2 subnets as shown in Fig. 1(a). A thorough description of the inter-subnet routing scheme is currently out of scope of the IBA specification and much freedom is given to the router vendors when implementing inter-subnet routing. The inter-subnet routing process defined in the IBA specification is similar to the routing in TCP/IP networks. First, if an end-node

want to send a packet to another subnet, the address resolution makes the local router visible to that end-node. The end-node puts the local router's LID address in the *local routing header* (LRH) and the final destination address (GID) in the *global routing header* fields. When the packet reaches a router, the packet fields are replaced (the source LID is replaced with the LID of the router's egress port, the destination LID is replaced with the LID of the next-hop port, and CRCs are recomputed) and the packet is forwarded to the next hop. The pseudo code for the rest of the packet relay model is described in [7] on page 1082. In this paper, we will only consider topologies similar to that presented in Fig. 1(a), i.e. cases where one or more subnets are directly connected using routers. Furthermore, each subnet must be a fat-tree topology and it must be directly attached to the other subnets without any transit subnets in between.

4 Layer-3 Routing in InfiniBand

Up until now, IB-IB routers were considered to be superfluous. Even the concept of *routing*, which in IP networks strictly refers to layer-3 routers, in IB was informally applied to forwarding done by layer-2 switches that process packets based only on their LID addresses. With the increasing size and complexity of subnets the need for routers has become more evident. There are two major problems with inter-subnet routing: which router should be chosen for a particular destination (first routing phase) and which path should be chosen by the router to reach the destination (second routing phase). Solving these problems in an optimal manner is not possible if adhering to the current IB specification: the routers are non-transparent subnet boundaries (local SM cannot see beyond), so full topology visibility condition is not met. However, in this paper, by using regularity features provided by the fat-tree topology, we propose a solution for these problems. Nevertheless, for more irregular networks where the final destination is located behind another subnet (at least two router hops required) there may be a need for a super subnet manager that coordinates between the local subnet managers and establishes the path through the transit subnet. We consider such scenarios to be future work. In this section we present two new routing algorithms: Inter-Subnet Source Routing (ISSR) and Inter-Subnet Fat-Tree Routing (ISFR). ISFR is an algorithm designed to work best on fat-trees while ISSR is a more generic algorithm that works well on other topologies also. However, in this paper we only focus on fat-trees and fat-tree subnets as the deadlock problem becomes more complex when dealing with irregular networks. Nevertheless, we plan to address deadlock free inter-subnet IB routing in a more general manner in subsequent publications.

4.1 Inter-subnet Source Routing

We designed ISSR to be a general purpose routing algorithm for routing hybrid subnets. It needs to be implemented both in the SM and the router firmware. It is a deterministic oblivious routing algorithm that always uses the same path

for the same pair of nodes. In general, it offers routing performance comparable to ISFR algorithm provided a few conditions explained in Sect. 5 are met.

The routing itself consists of two phases. First, for the local phase (choosing an ingress router port for a particular destination) this algorithm uses a mapping file. Whereas the $find\_router()$ function which chooses the local router looks almost exactly the same (it just matches the whole GID) for ISSR algorithm as for the OpenSM routing algorithm, the main difference lies in the setup of the mapping file. In our case, we provide full granularity meaning that instead of only a subnet prefix as for the OpenSM inter-subnet routing, the file now contains a fully qualified port GID. This means that we can map every destination end-port to a different router port while OpenSM routing can only match a whole subnet to a single router port. In the case of ISSR, an equal number of destinations is mapped to a number of ports in a round robin manner. In our example, *dst_gid1* and *dst_gid3* are routed through port 1 and port 2 on router A, and *dst_gid2* and *dst_gid4* are routed through the same ports on router B. Backup and default routes can also be specified.

Code 1. A high-level example of a mapping file for ISSR and ISFR algorithms

```
1: dst_gid1     router_A_port_1_guid
2: dst_gid2     router_B_port_1_guid
3: dst_gid3     router_A_port_2_guid
4: dst_gid4     router_B_port_2_guid
5: #default route
6:      *      router_A_port_1_guid
7:      *      router_B_port_1_guid
```

Second difference is the implementation of the additional code in the router firmware. A router receiving a packet destined to another subnet will source route that packet. The routing decision is based both on the source LID (of the original source or the egress port of the previous-hop router in a transit subnet scenario) and the destination LID (final destination LID or the LID of the next-hop ingress router port). The router knows both these values because it sees the subnets attached to it. To obtain the destination LID, a function mapping the destination GID to a destination LID or returning the next-hop LID based on the subnet prefix located in the GID is required. In our case, this function is named get_next_LID (line 2 of the pseudo code in Algorithm 1).

The algorithm calculates a random number based on the source and destination LIDs. This is done in a deterministic manner so that a given src-dst pair always generates the same number, which prevents out of order delivery when routing between subnets and, unlike round-robin, makes sure that each src-dst always uses the same path through the network. This number is used to select a single egress port from a set of possible ports. There is a set of possible ports because a router may be attached to more than two subnets and therefore a two-step port verification is necessary: first choose the ports attached to the subnet

Algorithm 1. choose_egress_port() function in ISSR

Require: Receive an inter-subnet packet
Ensure: Forward the packet in a deterministic oblivious manner
1: **if** *received_intersubnet_packet*() **then**
2: $dstLID = get\_next\_LID(dGID)$
3: $srand(srcLID + dstLID)$
4: $port\_set = choose\_possible\_out\_ports()$
5: $e\_port = port\_set[(rand()\%port\_set.size)]$
6: **end if**

(or in the direction of the subnet) in which the destination is located, and then, by using a simple hash based on a modulo function, choose the egress port.

4.2 Inter-subnet Fat-Tree Routing

As mentioned previously, a problem that needs to be solved is the communication between SMs that are in different subnets and are connected through non-transparent routers. Our solution is based on the fact that the IBA specification does not give the exact implementation for inter-subnet routing, so our proposal provides an interface in the routers through which the SMs will communicate. In other words, we implement handshaking between two SMs located in neighboring subnets. The algorithm uses the previously defined file format containing the GID-to-router port mappings in Code 1. The ISFR algorithm is presented in Algorithm 2. Like the ISSR algorithm, it is also implemented in the router device. ISFR works only on fat-trees and with fat-tree routing running locally in every subnet. It will fall back to ISSR if those conditions are not met.

Algorithm 2. query_down_for_egress_port() function in ISFR

Require: Local fat-tree routing is finished
Require: Received the mapping file
Ensure: Fat-tree like routing tables throughout the fabric
1: **if** *received_mapping_files* **then**
2: **for all** *port in down_ports* **do**
3: $down\_switch = get\_node(port)$
4: $lid = get\_LID\_by\_GID(GID)$
5: **if** $down\_switch.routing\_table[lid] == primary\_path$ **then**
6: $e\_port = port$
7: **end if**
8: **end for**
9: **end if**

Every single router in a subnet receives the port mappings from its local SM and is thereby able to learn which of its ports are used for which GIDs. Next, for each attached subnet, the router queries the switches in the destination subnets

to learn which of the switches has the primary path to that subnet's HCAs. If we assume a proper fat-tree (full bisection bandwidth) with routers on the top of the tree, then after such a query is performed, each router will have one path per port in the downward direction for each destination located in a particular subnet. In other words, if we substituted the top routers with switches, the routing tables for the pure fat-tree and the fat-tree with routers on top would be the same.

5 Simulations

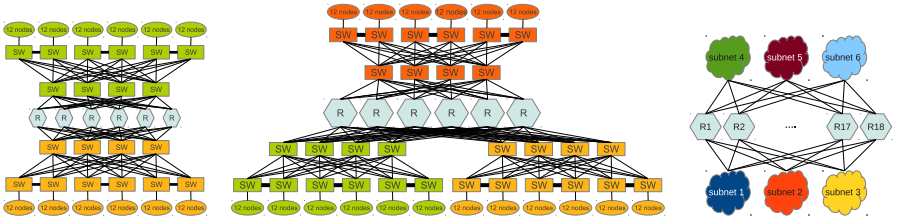
To perform large-scale evaluations and verify the scalability of our proposal, we use an InfiniBand model for the OMNEST/OMNeT++ simulator [13]. The IB model consists of a set of simple and compound modules to simulate an IB network with support for the IB flow control scheme, arbitration over multiple virtual lanes, congestion control, and routing using linear routing tables. In each of the simulations, we used a link speed of 20 Gbit/s (4x DDR) and Maximum Transfer Unit (MTU) equal to 2048 bytes. Furthermore, we use uniform, non-uniform and HPCC traffic patterns. We used synthetic traffic patterns to show baseline performance as these patterns have a predictable and easily understandable behavior, and are general rather than specific to a given application. We do not provide the baseline results for the same topologies implemented as a single subnet because ISFR routing provides exactly the same performance.

The simulations were performed on three different topologies shown in Fig. 1. Each of the topologies can be classified as a 3-stage (i.e. having three routing/switching stages and one node stage) fat-tree with routers placed on top of the tree (instead of normally placing root switches there). Even though there is a dedicated fat-tree routing algorithm delivering high performance on almost any fat-tree, we still decided to subnet a fat-tree fabric. The reason for that is that we consider the fat-tree topology to be a very good proof-of-concept topology for inter-subnet routing testing.

The fat-tree topology is scalable and by changing the number of ports we are able to vary the size of the topologies and show how our algorithms scale with regards to the number of nodes and subnets that are interconnected. All our subnets are 2-stage fat-trees that are branches in a larger 3-stage fat-tree so we can use routers and our routing algorithms to demonstrate how to seamlessly interconnect smaller fat-tree installations without using oversubscription. We chose a 3-stage 648-port fat-tree as the base fabric because it is a common configuration used by switch vendors in their own 648-port systems [14,15,16]. Additionally, such switches are often connected together to form larger installations like the JuRoPA supercomputer [17].

5.1 Routing Algorithm Comparison

We perform three sets of simulations: with uniform traffic, with non-uniform traffic and we run the HPCC benchmark. For non-uniform traffic we vary packet

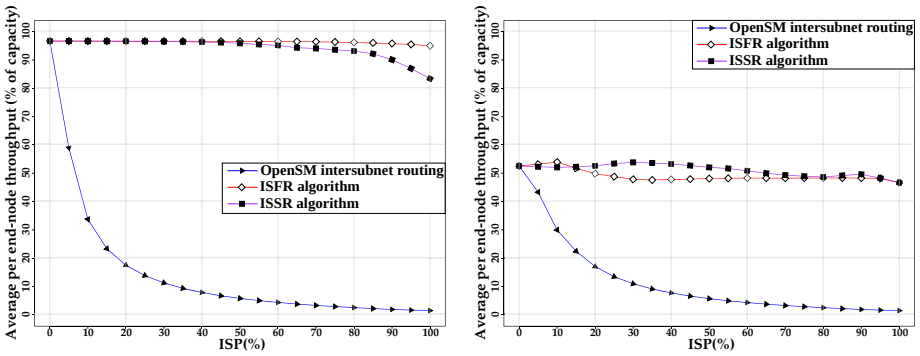


(a) 2-subnet fabric (b) 3-subnet fabric (c) 6-subnet fabric

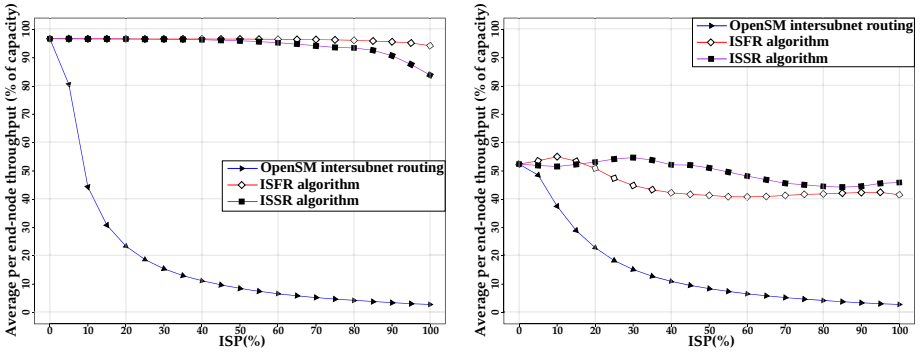
Fig. 1. Topologies used for the experiments

length from 84 bytes to 2 kB, keep the message length constant at 2 kB. We also introduce some randomly preselected hot spots (different for every random seed, not varying in time): one hot spot per subnet, with a probability of $ISP * 0.05$ for remote traffic and the same hot spot with a probability of $(1 - ISP) * 0.05$ for local traffic. The ISP (Inter-Subnet Percentage) value is the probability that a message will be sent to the local or the remote subnet. It varies from 0% (where all messages remain local) to 100% (where all messages are sent to remote subnets). The non-hot spot destined part of the traffic selects their destination randomly from all other available nodes. This means that there could be some other random hot spots that vary in time, and some nodes could also contribute unknowingly to the preselected hot spots. We express the measured throughput as the percentage of the available bandwidth for all the scenarios. The parameters for that traffic pattern were chosen to best illustrate the impact of congestion on the routing performance, which is a good baseline for algorithm comparison.

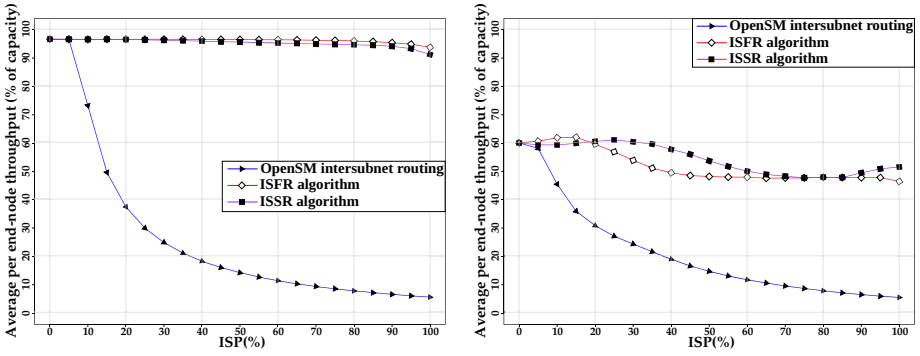
Uniform Traffic. The results for this scenario are shown in Fig. 2. When it comes to uniform traffic, we can establish that the performance of the OpenSM inter-subnet routing deteriorates in the presence of even a very small amount of inter-subnet traffic. At ISP equal to 20%, throughput is reduced to 17.5% for the 2-subnet scenario in Fig. 2(a), 23.46% for the 3-subnet scenario in Fig. 2(c), and 37.5% for the 6-subnet scenario in Fig. 2(e). The increase in performance for a larger number of subnets is explained by the fact that traffic is spread across more routers, i.e. each subnet in the topology uses a different ingress port locally. ISFR algorithm provides almost constant high performance under uniform traffic conditions whereas the performance of ISSR algorithm deteriorates slightly for very high ISP values as shown in Fig. 2(a) and Fig. 2(c). This is caused by the fact that egress ports from the routers may not be unique as they are chosen randomly. However, the deterioration is smaller when the number of subnets increases (12% decrease for the 3-subnet scenario compared to 5% decrease for the 6-subnet scenario at ISP=100%) as shown in Fig. 2(e). This occurs because the more subnets we have in the fabric and the higher is the ISP value, the more inter-subnet traffic pairs are created, so the hash function has a higher probability to utilize more links from the defined subnet-port-set as there are more random numbers generated.



(a) 2-subnet scenario uniform traffic (b) 2-subnet scenario non-uniform traffic



(c) 3-subnet scenario uniform traffic (d) 3-subnet scenario non-uniform traffic



(e) 6-subnet scenario uniform traffic (f) 6-subnet scenario non-uniform traffic

Fig. 2. Throughput as a function of ISP with uniform and non-uniform traffic

Non-uniform Traffic. Whereas under uniform traffic our algorithms gave almost optimal performance, the situation worsens if some non-uniformity is added as seen in Fig. 2.

What is first noticeable is the fact that ISFR algorithm is clearly outperformed by the ISSR algorithm for the middle range of the ISP values (20% to 70%), as

best seen in Fig. 2(d). Second, we observe that ISSR algorithm deteriorates for ISP values greater than 40%, which is best seen in Fig. 2(f). ISFR, on the other hand, becomes stable at ISP values close to 40%. This behavior is explained by the addition of the hot spots to our traffic pattern, the occurrence of head-of-line (HOL) blocking, and the migration of the root and the branches of the congestion trees. For lower ISP values ($\leq$50%) local traffic is dominant and in every subnet 5% of such traffic is destined to the local hot spot. In such a case the branches of the congestion tree will mostly influence the local traffic, but they will also grow through the single dedicated downward link (a thick branch) to influence the victim nodes in other subnets if the ISFR algorithm is used. For ISSR, the same will happen, but there is no dedicated downward link and the branches growing through multiple downward links will be much thinner, therefore, influencing the local traffic in other subnets to a lesser extent. This happens because ISSR spreads the traffic destined towards the hot spot across multiple downward links. This is the reason why ISFR algorithm is outperformed by ISSR in such a hot spot scenario for almost all ISP values.

For ISFR algorithm, for higher ISP values ($\geq$50%), the root of the congestion tree will move from the last link towards the destination to the first downward dedicated link towards the destination (i.e. a router port), and the congestion tree will influence mostly the inter-subnet traffic as there will be little or no local traffic, which is why ISFR algorithm reaches stability at around 50% ISP. For ISSR, for higher ISP values, the root of the congestion tree will not move and the branches will grow much thicker (as there is more incoming remote traffic). This will not only slightly influence the local traffic (that is low for high ISP values) in the contributor's subnets, but also it will cause HOL blocking for the downward traffic that uses the same links that the hot spot traffic uses to reach other destinations. It happens because ISSR does not use dedicated paths for downward destinations. Such a deterioration can be best observed in Fig. 2(d) or Fig. 2(f).

Another vital observation is the fact that by increasing the number of subnets, we increase the performance of all the routing algorithms. This is best visible when comparing Fig. 2(d) and Fig. 2(f). The explanation for that is the segmentation of the hot spot contributors. In other words, the more hot spots there are, the weaker is the influence of the head-of-line blocking (the congestion tree branches are thinner).

We also see that OpenSM routing still yields undesirable performance for every scenario. However, an important observation here is that the congestion does not originate from the hot spots, but from the utilization of a single ingress link to transmit the traffic to the other subnet.

HPC Challenge Benchmark. We implemented a ping-pong traffic pattern that was used to run the HPC Challenge Benchmark [18] tests in the simulator. We used a message size of 1954KB and kept the load constant at 100%. The tests were performed on 500 ring patterns: one natural-ordered ring (NOR) and 499 random-ordered rings (ROR) from which the minimum, maximum and average results were taken. In this test each node sends a message to its left neighbor in the ring and receives a message from its right neighbor. Next, it sends a message

Table 1. The HPC Challenge Benchmark results (in MB/s)

Measurement	OpenSM	ISSR	ISFR
NOR BW	1572.49	1572.49	1572.49
ROR BW min/max/avg			
2 subnets	528/878/703	847/1166/1001	1064/1314/1187
3 subnets	345/611/482	753/993/867	946/1165/1069
6 subnets	202/343/270	709/875/775	841/1018/933

back to its right neighbor and receives a return message from its left neighbor. We treated the whole fabric as a continuous ring and we disregarded the subnet boundaries.

Table 1 presents the HPCC Benchmark results. For any fat-tree the NOR bandwidth results give the maximum throughput as there is no contention in the upward or the downward direction. However, when we compare the results for the ROR, we observe differences between the routing algorithms. For the 2-subnet scenario, we observe an increase in throughput of 536 MB/s (102%) when comparing the minimum throughput for the OpenSM and the ISFR algorithms. Furthermore, we observe that the average throughput for the ISFR algorithm is higher than the maximum throughput for the ISSR algorithm in all cases. For the average throughput we observe an increase of 484 MB/s (69%) compared to OpenSM routing. For the 3-subnet scenario the trend is the same as for the previous scenario, but we observe that the throughput is lower than for the 2-subnet scenario. This happens because the larger the topology, the higher the probability that the destination is chosen from a set of non-directly connected nodes. For a 144-node fabric, each source can address 143 end-nodes and 23 out of those end-nodes (15.7%) are reachable through a non-blocking path (11 at the local switch, 12 at the neighbor switch). For a 216-node fabric, the same number of nodes is reachable through a non-blocking path, but the overall number of nodes is larger, which gives only 10.6% of nodes reachable through a non-blocking path. This means that a ROR pattern in a larger fabric has a lower probability for reaching a randomly chosen node in a non-blocking manner. Furthermore, in larger topologies more nodes are non-local, which means that the routing algorithm uses the longest hop path to reach them (traversing all stages in a fat-tree), which further decreases the performance. For the 6-subnet scenario, we observe a similar situation as for the 3-subnet scenario: that there is an overall decrease in performance. The explanation is the same as for the 3-subnet scenario: more nodes are used to construct a ROR, but the number of nodes accessible in a non-blocking manner stays the same, so the generated ROR pairs have an even lower probability to use a non-blocking path.

The general observation is that for the HPCC benchmark, the ISFR algorithm delivers the best performance. It is because this traffic pattern does not create any destination hot spots and the congestion occurs only on the upward links towards the routers, while the dedicated downward paths are congestion-free. Despite using the same upward path as the ISFR algorithm, ISSR algorithm may

not provide a dedicated downward path, which is why there is a performance difference between these two algorithms.

6 Conclusions and Future Work

Native IB-IB routers will make the network scalable, and designing efficient routing algorithms is the first step towards that goal. In this paper, we laid the groundwork for layer-3 routing in IB and we presented two new routing algorithms for inter-subnet routing: the inter-subnet source routing and the inter-subnet fat-tree routing. We showed that they dramatically improve the network performance compared to the current OpenSM inter-subnet routing.

In future, we plan to generalize our solution to be able to support many different regular fabrics in a deadlock-free manner. Another candidate for research will be evaluating the hardware design alternatives. Looking further ahead, we will also propose a deadlock-free all-to-all switch-to-switch routing algorithm.

References

1. Obsidian Strategics: Native InfiniBand Routing (2008), http://www.nsc.liu.se/nsc08/pres/southwell.pdf
2. Southwell, D.: Next Generation Subnet Manager - BGFC. In: Proceedings of HPC Advisory Council Switzerland Conference 2012 (2012)
3. InfiniBand Trade Association: Introduction to InfiniBand for End Users (2010)
4. Obsidian Strategics: Native InfiniBand Routing (2006), http://www.obsidianresearch.com/archives/2006/Mellanox_Obsidian_SC06_handout_0.2.pdf
5. The OpenFabrics Alliance: Issues for Exascale, Scalability, and Resilience (2010)
6. Yousif, M.: Security Enhancement in InfiniBand Architecture. In: 19th IEEE International Parallel and Distributed Processing Symposium, pp. 105a. IEEE (April 2005)
7. InfiniBand Trade Association: Infiniband Architecture Specification, 1.2.1 edn. (November 2007)
8. Prescott, C., Taylor, C.: Comparative Performance Analysis of Obsidian Longbow InfiniBand Range-Extension Technology (2007)
9. Richling, S., Kredel, H., Hau, S., Kruse, H.G.: A long-distance InfiniBand interconnection between two clusters in production use. In: State of the Practice Reports on - SC 2011. ACM Press, New York (2011)
10. Top 500 Supercomputer Sites (November 2012), http://top500.org/
11. Dally, W.J., Towles, B.: Principles and practices of interconnection networks. Morgan Kaufmann (2004)
12. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks an Engineering Approach. Morgan Kaufmann (2003)
13. Gran, E.G., Reinemo, S.A.: Infiniband congestion control, modelling and validation. In: 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools 2011, OMNeT++ 2011 Workshop) (2011)
14. Oracle Corporation: Sun Datacenter InfiniBand Switch 648, http://www.oracle.com/us/products/servers-storage/networking/infiniband/046267.pdf

15. Mellanox Technologies: Voltaire Grid Director 4700, http://www.voltaire.com/assets/files/Datasheets3/Grid-Director-4700-DS-WEB-020711.pdf
16. Mellanox Technologies: IS5600 - 648-port InfiniBand Chassis Switch, http://www.mellanox.com/related-docs/prod_ib_switch_systems/IS5600.pdf
17. Forschungszentrum Jülich: JuRoPA - Jülich Research on Petaflop Architectures
18. Luszczek, P., Dongarra, J.J., Koester, D., Rabenseifner, R., Lucas, B., Kepner, J., McCalpin, J., Bailey, D., Takahashi, D.: Introduction to the HPC Challenge Benchmark Suite (April 2005)