Routing and Fault-Tolerance Capabilities of the Fabriscale FM compared to OpenSM

Jesus Camacho Villanueva, Tor Skeie, and Sven-Arne Reinemo

Fabriscale Technologies

E-mail: {jesus.camacho,tor.skeie,sven-arne.reinemo}@fabriscale.com

July 11, 2015

Abstract

Fabriscale is a new generation user friendly and reliable fabric manager for InfiniBand. In this white paper we demonstrate Fabriscale's routing and fast fault-tolerance capabilities. First we assess its routing quality by studying agnostic routing of random topologies, as a fundament for plug & play networking. We consider fat-trees too as the most widespread used topology, also when they are subject to failures. Herein we compare the performance of Fabriscale with the routing engines offered by OpenFabrics' OpenSM through simulations - the results show that Fabriscale outperforms existing OpenSM routing with 13%-80%, depending on the case under inspection. Next we show that Fabriscale offers dynamic fast-tolerance with no need for complete reconfiguration (heavy sweep) of the fabric. It switches to alternative paths in just a few milliseconds time reducing application downtime to a minimum, herein a reduced chance for abnormal application termination. The results presented in this paper show that Fabriscale offers optimized performance and fault-tolerance compared to OpenSM, and makes it suitable for fail-in-place network design.

1 Fabriscale FM features

Fabriscale is a new generation fabric manager that ensures more efficient and reliable operation of Data Centre's and High Performance Computing clusters. Fabriscale currently offers the following unique features:

Network agnostic operation: Fabriscale works out-of-the-box and supports plug-and-play networking independently of the network layout.

Supreme fault-tolerance: Fabriscale switches to alternative paths in just a few milliseconds (no need for heavy sweep), and can in this way offer fast fail-over reducing application downtime to a minimum. Redundant paths are computed upfront providing graceful degradation and assures that applications will be minimally affected by network problems, which opens up for fail-in-place network design (no need to fix/replace faulty components straight way).

Virtual layer scalability: Fabriscale has the ability to scale with respect to the number of virtual links (layers) made available, where this scalability is two-fold:

- Fabriscale will always offer deadlock-free routing and work out-of-the-box even if only one single virtual layer is available. Under those circumstances for instance the OpenSM's (agnostic) routing engines fail to route the fabric for arbitrary topologies, as well as when larger fat-trees contain failures.
- As more and more virtual layers are made available the performance of Fabriscale increases.

Software defined networking (SDN) interface: Fabriscale allows applications to configure the network according to their needs without human intervention.

Modern web-based GUI: Makes it easy for the user to monitor the fabric for faults and various performance metrics. Extendable using the REST API.

The SDN and GUI features will be presented in a separate document available on Fabriscale's web-page.

2 Network agnostic operation

Fabriscale can handle any network topology and in that respect will always work out-of-the-box. Agnostic routing is the fundament for achieving this feature including plug & play networking. OpenSM currently offers three topology agnostic routing engines known as Up/Down [1], LASH [2, 3], and DFSSSP [4]. Those algorithms have different strengths and weaknesses. Up/Down is simple in the way that it only needs a single virtual layer in order to be deadlock-free as it is based on spanning-tree routing, however, its performance is rather poor since links in the proximity of the root tend to become bottlenecks. LASH on the other hand offers shortest-path routing and better performance (not restricted to spanning-tree routing), but needs virtual layers and the number of virtual layers required increases with the size of the topology (for random topologies). The third alternative DFSSSP is an improvement of LASH, opposed to LASH it balances the use of the network resources very well and outperforms LASH with respect to performance - however, it consumes virtual layers, as LASH does, in order to guarantee deadlock-free routing. As a consequence both LASH and DFSSSP fail to route the fabric if required virtual lanes is higher than the number of available virtual lanes. For a thorough study on topology agnostic routing algorithms, see [5].

Below we present simulation results of random topologies to study the network agnostic operation capability. As concerns reasonable sized random topologies we consider networks consisting of 128 and 256 switches, where the number of links is twice the number of switches that is the most severe configuration to handle from a deadlock routing point of view [2]. For all the test cases Fabriscale is being compared to the OpenSM routing engines. Before presenting those performance results we will take you through the test-bed based on OpenSM and OMNeT++ simulation environment.

2.1 The Fabriscale-OMNeT++ test-bed

To perform large-scale evaluations and verify the scalability of our proposal, we use an InfiniBand model for the OMNeT++ simulator [6]. The model contains an implementation of IB HCAs and switches with support for the IB flow control scheme, arbitration over multiple virtual lanes, and routing using linear forwarding tables. The communication links are 4x SDR, which have the maximum effective throughput of 8Gbit/s - we assume 8b/10b encoding so the maximum gross throughput is 10Gbit/s. The topology is simulated using the *ibsim* management simulator from OpenFabrics and routing tables are generated using the Fabriscale fabric manager. These routing tables are then converted into OMNeT++ readable format in order to simulate network traffic on real-world systems.

2.2 Performance of random network topologies

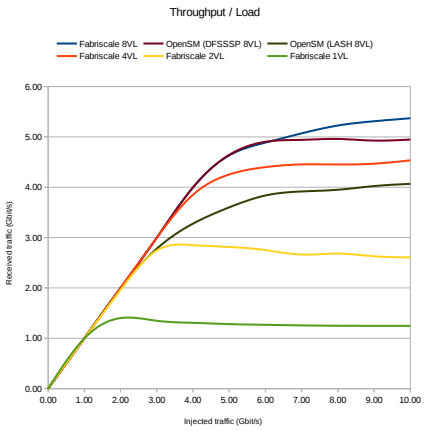
Figure 2 depicts the throughput for two sizes of random (irregular) network sizes consisting of 128 and 256 switches, respectively. The plots give the average value of 10 different randomly generated topologies, and where the network traffic is modelled as a uniform random communication pattern.

The Fabriscale graphs herein show the virtual layer scalability property of Fabriscale: i) being able to offer deadlock-free routing even if only single virtual layer (VL) is available. ii) As more virtual layers are made available the throughput of Fabriscale increases accordingly (being tested with 1, 2, 4, and 8 VLs). This is, however, not the case for the comparable OpenSM routing engine's - for the 128 switch networks both LASH and DFSSSP requires more than 4 VLs to provide deadlock-free routing (figure 1(a)). Performance-wise Fabriscale also surpasses the OpenSM algorithms (the 8 VL configuration). When the network size is increased to 256 switch networks (figure 1(b)), in fact both LASH and DFSSSP fails to route the fabric (the number of VLs is limited to 8), while Fabriscale on the other hand provides deadlock-free routing for any number of virtual layers.

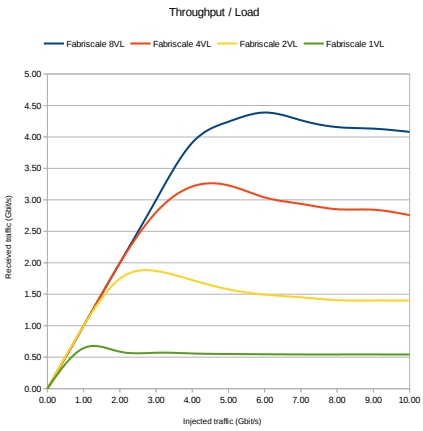
3 Routing of Fat-trees

Fabriscale's routing of fat-trees is a slightly modified version of the agnostic routing algorithm described above that is optimised for routing of fat-trees. For a formal description of fat-trees (k-ary n-cubes), see [7]. When the topology discovery mechanism detects the network as a fat-tree this specific routing algorithm will be called on. By optimised we mean that we balance the link usage in the downward routing phase so that there is no contention assuming uniform distributed traffic (each downward link just carries traffic to one destination).

The fat-tree routing engine also computes disjoint redundant paths upfront, where those paths are carefully selected to offer marginal degradation in presence of failures, even if multiple faults occur. The redundant paths will be activated by the Fabriscale FM momentarily (only a few milliseconds delay) without the need to recalculate the forwarding tables. Below we present simulation results of different fully populated k-ary n-trees (without failures) where we compare the performance of Fabriscale to the following OpenSM's routing engines: i) the fat-



(a) Networks consisting of 128 switches.



(b) Networks consisting of 256 switches - only Fabriscale offers solutions for this configuration, when number of virtual layers is limited to 8.

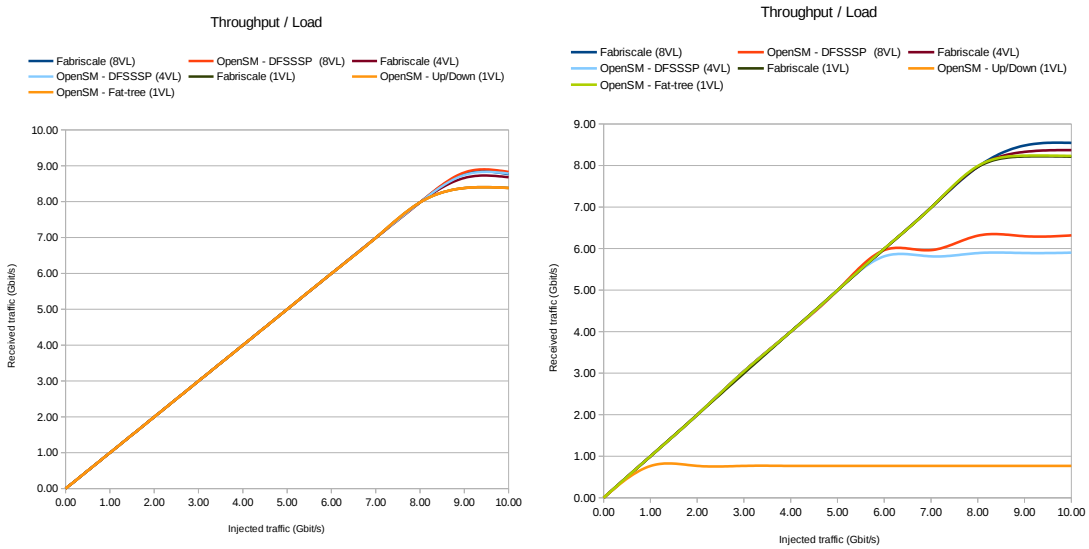
Figure 1: *Average throughput of 10 different random network topologies, given a uniform traffic pattern.*

tree algorithm, which is specialised for fat-trees being perfect balanced for uniform traffic. ii) the Up/Down algorithm, which is topology agnostic being spanning-tree based. It has, however, been refined to be better suited for fat-trees where multiple roots can be specified (e.g. to match the number of top-level switches in the fat-tree). iii) the DFSSSP algorithm, which also is agnostic as discussed above. DFSSP is interesting from the point of view it has shown good performance figures also for fat-trees both without and with failures [8].

3.1 Performance of fat-trees

Figure 2(a) depicts the throughput of an 18-ary two level fat-tree consisting of 324 terminal ports and 27 switches. It is a well known configuration and building block being productized, as for instance the SX6536 switch from Mellanox [9]. This (fault-free) network configuration is fully connected by means of that every switch on the lowest level has a link to each switch on the second level (and vice versa), which makes it rather straightforward to achieve a perfectly balanced routed network, given a uniform traffic pattern. This is confirmed by the shown performance curves, where the throughput is close to optimal for both OpenSM and Fabriscale. The only exceptions are the OpenSM's Fat-tree and Up/Down algorithms, both having the ex-

act same throughput, but are lagging a bit behind the other methods. Common for those two methods is that they are not able to scale with respect to the number of virtual layers available, and make use of 1 VL only. Fabriscale though achieves higher throughput than Fat-tree and Up/Down for the 1 VL configuration. The reason for that Up/Down achieves the same performance as the Fat-tree algorithm in a fully connected 2-level fat-tree is that the routing is straightforward (all paths are evenly used, just with one root).



(a) 18-ary 2-tree.

(b) 12-ary 3-tree.

Figure 2: *Throughput of fully connected fat-trees, given a uniform traffic pattern.*

We also consider a 3-level fat-tree, the 12-ary 3-tree that consists of 1728 terminal ports and 360 switches. The average throughput results are shown in figure 2(b). We see here that both OpenSM's Fat-tree algorithm and Fabriscale are close to optimal due to perfect traffic balancing (though, Fabriscale gets a slight boost when adding VLS). On the other hand OpenSM's DFSSSP algorithm achieves far less throughput for the 3-level fat-tree compared to Fabriscale (surpassed by around 40%). The Up/Down algorithm seems to be completely useless for 3-level fat-trees because of very poor balancing.

4 Fault-tolerance capabilities

In this section we demonstrate Fabriscale's ability to offer dynamic and fast fault-tolerance as an instrument to:

- Ensure that applications will be minimally affected by network problems due to Fabriscale's graceful degradation in presence of faults.
- Reduce the application downtime to a minimum by offering quick fail-over.

As already mentioned changes in the network topology (when having addition or removal of links/switches) do not enforce a complete reconfiguration of the fabric. Fabriscale makes use of topologies' path multiplicity along with precomputed alternative routes to implement quick fail-over (migration). The quick fail-over mechanism assures that applications do not terminate abnormally during the fail-over phase regardless of the topology size, in contrast to OpenSM.

Below we present results from performance experiments to throw light on Fabriscale's fault-tolerance capabilities: First, we will show the quick fail-over behaviour of Fabriscale, verifying that it scales. Secondly, we demonstrate Fabriscale's graceful degradation in presence of faults for different sizes of fat-trees conducted as simulations (where performance is expressed as network utilisation). For all the experiments we compare the performance of Fabriscale to the OpenSM's routing engines.

4.1 Fail-over times

We here study the fail-over times of Fabriscale and OpenSM, defined as the time between the subnet manager has been notified that a port (link) is down (on reception of traps) and the fabric is connected again for all communicating pairs and the application can continue running (if it has not crashed). For Fabriscale this is managed by activating redundant paths, while for OpenSM a complete recalculation and distribution of paths (heavy sweep) will be undertaken. In order to be able to conduct large scale experiments we have used OpenFabrics' management tool IBSim for this purpose. IBSim is realistic in the sense that it models all the management behaviour related to faults occurring (topology changes) except that time to program the routing tables has been observed to be slightly lower than the time to program real switches. Thus, the fail-over times we present here will be slightly lower than what you will experience in real life.

We have studied the fail-over times for 18-ary two and three level fat-trees representing realistic network sizes, consisting of 324 and 5832 nodes, respectively. Table 1 shows the average fail-over times of 10 different link fault situations for each of the referred topologies. As seen the fail-over time for the 2-level case is quite short for all the algorithms, where it takes 120 millisec. for DFSSSP, 39 millisec. for fat-tree, and 14 millisec. for Fabriscale. Most applications will tolerate those fail-over times without abnormal termination. For the 3-level fat-tree we do, however, see a huge increase in the fail-over times for DFSSSP and fat-tree reaching 144 sec. and 104 sec., respectively. Under normal circumstances those long fail-over times will most likely cause applications to terminate abnormally. The fail-over time for Up/Down is slightly below one minute, which is on the edge what applications tolerate before aborting. Note that Up/Down does not implement sophisticated traffic balancing and in that respect needs less time to calculate the routing tables compared to the DFSSSP and Fat-tree algorithms. For Fabriscale on the other hand it takes only 0.35 sec. in average to reestablish connectivity, which is far below the time out values of most applications.

4.2 Performance of fat-trees with faulty links

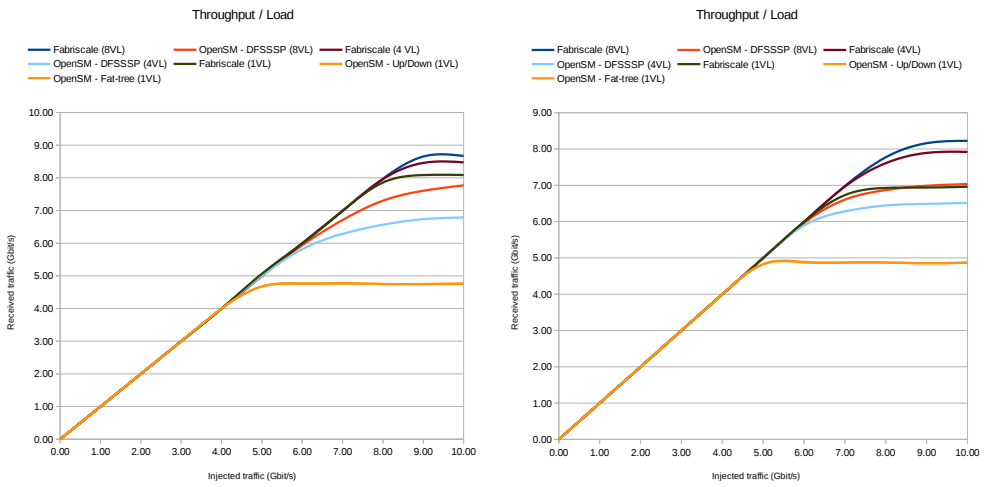
Let us now turn the attention to operational situations when the fat-tree is subject of failures and study how this affect the performance. Figure 3(a) depicts the throughput of a 18-ary 2-tree when 1% of the links are faulty at the same time. The plots show that Fabriscale has the far

Topology	*DFSSSP*	*Fat-tree*	*Up/(Down*	*Fabriscale*
18-ary 2-tree (324 nodes)	0.120	0.039	0.040	0.014
18-ary 3-tree (5832 nodes)	144	104	54	0.34

Table 1: *The fail-over times (in seconds) for OpenSM's routing engines and Fabriscale for 2- and 3-level fat-trees built from 36 port switches.*

highest throughput, and outperforms existing OpenSM routing with 13%-80%. We can see that 4 VLs is sufficient to achieve this maximum throughput (i.e. 8 VLs does not give much gain beyond 4 VLs), it does, however, get a slight boost when going from 1 to 4 VLs. It is quite striking though that Fabriscale restricted to 1 VL achieves higher throughput than DFSSSP using 8 VLs.

Continue further down this path where as much as 3% of the links are faulty, those results are shown in figure 3(b). We roughly see the same picture here as for the 1% case where Fabriscale has the highest throughput (no matter whether 1 to 8 VLs is supported), and outperforms OpenSM routing with 18%-70%. OpenSM and Fabriscale is based on different approaches in presence of faults - DFSSSP and the Fat-tree algorithm have to recalculate their forwarding tables for each single fault (topology change) happening, while Fabriscale on the other hand make use of precomputed redundant paths without reconfiguration. Still Fabriscale achieves far higher throughput.



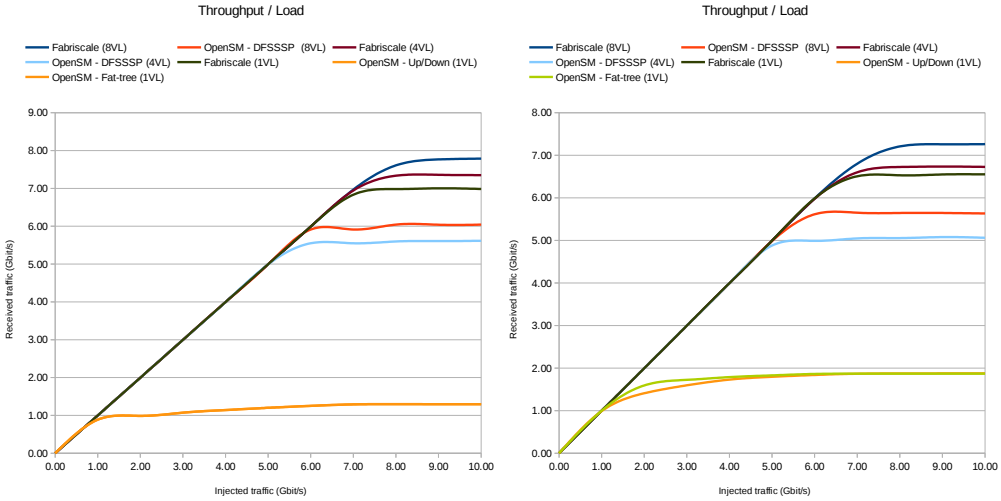
(a) 1% link failures.

(b) 3% link failures.

Figure 3: *Throughput of an 18-ary 2-tree in presence of link failures, given a uniform traffic pattern.*

Let us then look at the performance for a 3-level fat-tree in presence of faults, figure 4(a) shows the throughtput when a 12-ary 3-tree is subject to 1% link failures. We see the same

picture as for the 2-level fat-tree where Fabriscale outperforms OpenSM with around 30% for that configuration; and even with 1 VL Fabriscale achieves higher throughput than DFSSSP using 8 VLs. Note also that the Fat-tree algorithm is not capable of handling faults in 3-level fat-trees and becomes restriced to Min-hop routing. Introducing 3% link failures the tendency is confirmed, where Fabriscale performs around 24% better than OpenSM (4(b)).



(a) 1% link failures.

(b) 3% link failures.

Figure 4: *Throughput of a 12-ary 3-tree in presence of link failures, given a uniform traffic pattern.*

5 Summary

The Fabriscale fabric manager is a new generation of fabric management that ensures more efficient and reliable operation of data centres and high performance computing clusters based on InfiniBand technology. In this white paper show that Fabriscale offers supreme performance and fault-tolerance compared to OpenSM, and makes it suitable for fail-in-place network design.

Key benefits of the Fabriscale FM are:

- Ensures that applications will be minimally affected by network problems due to Fabriscale's graceful degradation in presence of faults. For fat-trees Fabriscale outperforms existing OpenSM routing with 13%-80%, depending on the case under inspection. In fact when Fabriscale is restricted to use only 1 VL it achieves higher throughput than OpenSM using 8 VLs.

- Reduces the application downtime to a minimum by offering quick fail-over, with no need for complete reconfiguration (heavy sweep) of the fabric. It switches to alternative paths

in just a few milliseconds time, and assures that applications do not terminate abnormally during the fail-over phase regardless of the topology size, in contrast to OpenSM.

- Offers virtual layer scalability being able to scale with respect to the number of virtual links (layers) made available, where this scalability is two-fold: Firstly, Fabriscale will always offer deadlock-free routing and work out-of-the-box even if only one single virtual layer is available. Secondly, as more and more virtual layers are made available the performance of Fabriscale increases.
- Offers a Software defined networking (SDN) interface, which allows applications to configure the network according to their needs without human intervention.
- Offers a modern web-based GUI, which makes it easy for the user to monitor the fabric for faults and various performance metrics. Extendable using the REST API.

References

[1] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, and T. L. Rodeheffer, *Autonet: a high-speed, self-configuring local area network using point-to-point links*, IEEE Journal on Selected Areas in Communications, vol. 9, no. 8, october 1991.

[2] T. Skeie, O. Lysne, and I. Theiss. *Layered shortest path (LASH) routing in irregular system area networks*, in Procedings of the 2002 International Parallel and Distributed Processing Symposium (IPDPS 2002), 2002, workshop 9: Communication Architecture for Clusters (CAC).

[3] O. Lysne, T. Skeie, S.A. Reinemo, and I. Theiss, *Layered routing in irregular networks*, IEEE Transactions on Parallel Distributed Systems, vol. 17, no. 1, pp. 51-65, 2006.

[4] J. Domke, T. Hoefler, and W. Nagel, *Deadlock-Free Oblivious Routing for Arbitrary Topologies*, In Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS), presented in Anchorage, AL, USA, pages 613–624, IEEE Computer Society, ISBN: 0-7695-4385-7, May 2011.

[5] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. Sancho, *A Survey and Evaluation of Topology Agnostic Routing Algorithms*, IEEE Transactions on Parallel and Distributed Systems 23 (2012): 405-425.

[6] E.G. Gran and S.A. Reinemo, *InfiniBand Congestion Control, Modelling and Validation*, In 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools2011, OMNeT++ 2011 Workshop). SIMUTools '11. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2011.

[7] F. Petrini and M. Vanneschi, *k-ary n-trees: High performance networks for massively parallel architectures*, in Proceedings of the 11th International Parallel Processing Symposium, 1997, IEEE, pp. 87-93.

[8] J. Domke, T. Hoefler, S. Matsuoka, *Fail-in-Place Network Design: Interaction between Topology, Routing Algorithm and Failures*, presented in New Orleans, LA, USA, Nov. 2014, accepted at IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC14).

[9] SX6536 switch from Mellanox, `http://www.mellanox.com/page/products_dyn?product_family=122&mtag=sx6536`