High-Quality Fault-Resiliency in Fat-Tree Networks (Extended Abstract)

John Gliksberg, Antoine Capra, Alexandre Louvet, Pedro Javier Garcia, Devan Sohier

High-Quality Fault-Resiliency in Fat-Tree Networks *(extended abstract)*

John Gliksberg*†‡, Antoine Capra†, Alexandre Louvet†, Pedro Javier García‡, and Devan Sohier*

* *Li-Parad, UVSQ, Versailles, France,* {`john.gliksberg`, `devan.sohier`} `@uvsq.fr`
† *BXI, Atos, Bruyères-le-Châtel, France,* {`antoine.capra`, `alexandre.louvet`} `@atos.net`
‡ *RAAP, UCLM, Albacete, Spain,* `pedrojavier.garcia` `@uclm.es`

Abstract*—Coupling regular topologies with optimized routing algorithms is key in pushing the performance of interconnection networks of HPC systems. In this paper we present Dmodc, a fast deterministic routing algorithm for Parallel Generalized Fat-Trees (PGFTs) which minimizes congestion risk even under massive topology degradation caused by equipment failure. It applies a modulo-based computation of forwarding tables among switches closer to the destination, using only knowledge of subtrees for pre-modulo division. Dmodc allows complete re-routing of topologies with tens of thousands of nodes in less than a second, which greatly helps centralized fabric management react to faults with high-quality routing tables and no impact to running applications in current and future very large-scale HPC clusters. We compare Dmodc against routing algorithms available in the InfiniBand control software (OpenSM) first for routing execution time to show feasibility at scale, and then for congestion risk under degradation to demonstrate robustness. The latter comparison is done using static analysis of routing tables under random permutation (RP), shift permutation (SP) and all-to-all (A2A) traffic patterns. Results for Dmodc show A2A and RP congestion risks similar under heavy degradation as the most stable algorithms compared, and near-optimal SP congestion risk up to 1% of random degradation.

1. Introduction

A majority of current leading network topologies for High Performance Computing (HPC) clusters are fat-tree variants. (The five most powerful clusters of the November 2018 Top500 list [1] boasted fat-tree topologies.) It is sufficient for fat-tree-specific routing algorithms to be minimal to guarantee deadlock-free routing, and the regular nature of their target topology class should simplify load-balancing strategies. In general, oblivious routing (without knowledge of communication patterns) in fat-trees is deterministic and optimized for shift patterns [2] [3] [4]. In particular, PGFTs [5] describe all regular fat-trees for which there is at most one downward switch-path from any switch to any node (as shown in Figure 1). The oblivious algorithm for non-degraded PGFTs (Dmodk) uses this property and their connection logic to provide load balance through an arithmetic rule.

Section 2 goes into detail about existing works for fault-resilient fat-tree routing and their shortcomings. The proposed fault-resilient routing algorithm is presented in Section 3. Section 4 then presents results by comparing our implementation against routing engines of OpenSM [6] (those deemed appropriate for routing of degraded fat-trees), first for routing runtime and then for congestion risk under degradation. Section 5 summarizes pros and cons of the proposed technique.

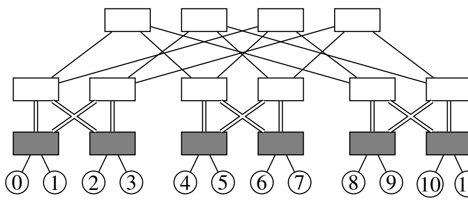

Figure 1. $PGFT(3; 2,2,3; 1,2,2; 1,2,1)$ with leaf switches shown in grey.

2. Background

Some of the research regarding oblivious fault-resilient routing focuses on techniques that apply to any connected network [7] [8]; these topology-agnostic techniques require full re-routing on topology change. Some other research explicitly targets degradations to regular fat-trees [4] [9]; there are several re-routing strategies for these techniques. OpenSM's UPDN [10] and Ftree [3] routing engines can also be applied from scratch to a degraded fat-tree. PQFT [4] is similar, though it requires a complete list of faults. The combination of Dmodk + Ftrnd_diff [11] available in BXI FM [12] is applied in an offline/online manner (with an iterative list of topology changes and an up-to-date view of the topology), the goal being fast reaction to faults with minimal routing changes. Fabriscale [13] also provides fast centralized re-routing of fat-trees, by precomputing alternative routes.

The random operation chosen in Ftrnd_diff results in progressive degradation of load balance and incapacity to return to the original routing in case of fault recovery. Ftrnd_diff does manage to recover rapidly from minor failures; however large numbers of simultaneous changes (which happen for example when entire islets are rebooted) cause computation times to skyrocket in current implementations. The strategies of PQFT and Fabriscale which consist in moving only invalidated routes let one expect somewhat similar load-balancing issues as with Ftrnd_diff. Studies show topology-agnostic routing outperforms fat-tree-specific routing under sufficient topology degradations [14] [8].

3. Dmodc Description

The idea behind the fault-resilient algorithm that we propose in this paper is to rely on local information like Ftree (and no topological address), while using the same closed-form arithmetic operation as Dmodk. The aim is fast centralized computation of routing tables for degraded PGFTs, providing optimal or well-balanced deterministic routes even under heavy fabric degradation. The algorithm begins with a partly sequential preprocessing phase followed by a parallel computation phase. Links are bi-directional; notations used in the expressions below are defined in Table 1.

TABLE 1. NOTATIONS USED IN EXPRESSIONS.

S	is the set of switches,
L	is the set of leaf switches ($L \subset S$),
$U(s)$	is the UUID of switch s,
N	is the set of nodes,
λ_n	is the (only) leaf switch connected to node n ($\lambda_n \in L$)
$\rightharpoondown, \rightharpoonup$	respectively denote downlinks and uplinks (according to rank),
G_s	is the ordered list of port groups of switch s,
Ω_g	is the switch connected to port group g,
$\#$	denotes cardinality, in number of port groups or of ports.
$c_{s,l}$	is the cost of switch s to leaf switch l,
Π_s	is the divider of switch s,
t_n	is the topological node identifier (NID) of node n,

3.1. Preprocessing

Rank. Levels and link directions are determined based on leaf switches being equivalent to the lowest level.

Port Groups. Groups of ports linked to the same switch are prepared and sorted by universally unique identifier (UUID, defined at hardware fabrication) to help with same-destination route coalescing.

Cost. We define the cost $c_{s,l}$ of a switch s to a leaf switch l to be the minimum number of hops between one another under up–down restrictions according to rank, as defined in Algorithm 1.

Divider. Dmodc is based on the same arithmetic operation as Dmodk: it begins with an integer division by the product of upward arities ($= \#\{s' \rightharpoondown s\}$) of lower levels. To reflect the actual state of the topology, only local information must be considered; in turn this operation is based on the products of up-to-date counts of upswitches, as defined in Algorithm 1.

Algorithm 1: Compute costs and dividers.

- **foreach** $s \in S$ **do**
  - **foreach** $l \in L$ **do** $c_{s,l} \leftarrow \infty$
  - $\Pi_s \leftarrow 1$
- **foreach** $l \in L$ **do** $c_{l,l} \leftarrow 0$
- **foreach** $s \in S$ *going upwards* **do**
  - $\pi \leftarrow \Pi_s \times \#\{s' \rightharpoondown s\}$
  - **foreach** $s' \rightharpoondown s$ **do**
    - **foreach** $l \in L \mid c_{s,l} + 1 < c_{s',l}$ **do**
      - $c_{s',l} \leftarrow c_{s,l} + 1$
    - **if** $\pi > \Pi_{s'}$ **then** $\Pi_{s'} \leftarrow \pi$ (max reduction)
- **foreach** $s \in S \setminus L$ *going downwards* **do**
  - **foreach** $s' \rightharpoonup s$ **do**
    - **foreach** $l \in L \mid c_{s,l} + 1 < c_{s',l}$ **do**
      - $c_{s',l} \leftarrow c_{s,l} + 1$

In a full PGFT, $\#\{s' \rightharpoondown s\}$ values are constant throughout each level, and individual values are decreased following faults; the $\max$ reduction accordingly helps keep load repartition stable under few faults. This choice of reduction was only compared with one using the first downward path and showed little to no change in route quality under random degradation.

Topological NID. The arithmetic nature of Dmodc guarantees load-balancing only if NIDs (on which the modulo operation is applied) are topologically contiguous. We explicitly determine each node's topological NID using previously computed costs in Algorithm 2.

Algorithm 2: Compute topological NIDs.

- $t \leftarrow 0$
- $X \leftarrow L$ sorted by UUIDs
- **while** $X \neq \emptyset$ **do**
  - $l \leftarrow X_0$
  - $\mu \leftarrow \min_{l' \in X \setminus \{l\}}(c_{l,l'})$
  - **foreach** $l' \in X \mid c_{l,l'} \leq \mu$ **do**
    - **foreach** $n \rightharpoonup l'$ *in port rank order* **do**
      - $t_n \leftarrow t$
      - $t \leftarrow t + 1$
      - $X \leftarrow X \setminus \{l'\}$

Dmodc can provide optimal results for shift permutation communication patterns which respect such an ordering, otherwise one could expect results similar to those of random permutation communication patterns.

3.2. Routes Computation

The deterministic output port $p_{s,d}$ and alternative output ports $P_{s,d}$ of every switch s for every destination $d \in N$ (not directly linked to s) are selected with a closed-form operation based on the results previously determined. First, port groups leading *closer* to λ_d are selected in (1), setting corresponding alternative output ports in (2):

$$C_{s,\lambda_d} \leftarrow \{g \in G_s \mid c_{\Omega_g,\lambda_d} < c_{s,\lambda_d}\} \quad (1)$$

$$P_{s,d} \leftarrow \{p \in g \mid g \in C_{s,\lambda_d}\} \quad (2)$$

Selected port groups $C_{s,\lambda_d}[i] \; \forall \; i \in \#C_{s,\lambda_d}$ are ordered by UUID of their remote switch. From this, the output port group is chosen in (3) and the port within that group in (4):

$$g_{s,d} \leftarrow C_{s,\lambda_d}\left[\left\lfloor \frac{t_d}{\Pi_s} \right\rfloor \bmod \#C_{s,\lambda_d}\right] \quad (3)$$

$$p_{s,d} \leftarrow g_{s,d}\left[\left\lfloor \frac{t_d}{\Pi_s \times \#C_{s,\lambda_d}} \right\rfloor \bmod \#g_{s,d}\right] \quad (4)$$

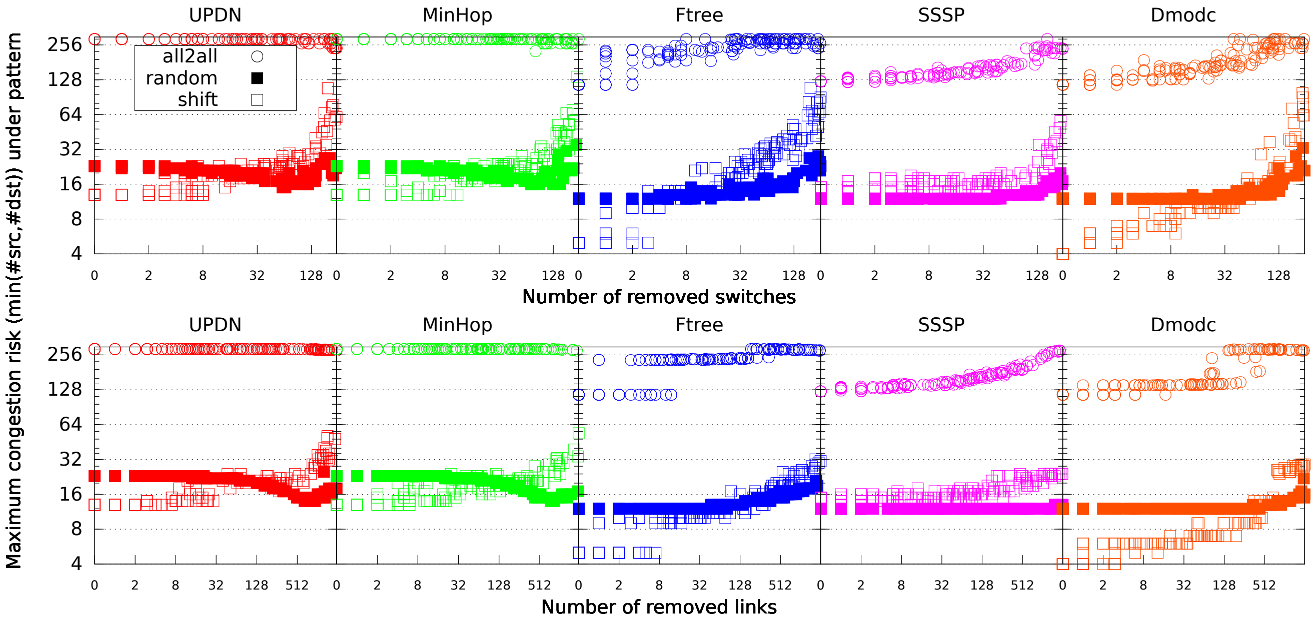

Figure 2. Maximum congestion risk in a 8640 node PGFT (with blocking factor of 4) under random topology degradation (in log–log scale; lower is better).

4. Results

Validity

Routing is valid for degraded PGFTs if and only if the cost of every leaf switch to every other leaf switch is finite: this reflects every node pair having an up–down path. Our implementation includes a pass through all leaf switch pairs to verify this condition. The up–down path restriction is sufficient to guarantee deadlock-freedom within degraded PGFTs [9].

Congestion Risk

Random degradation is simulated using hundreds of throws for each considered routing algorithm and type of equipment to degrade (switches or links). The integer amount of equipment $a \in [0, 2^m[$ to remove at each throw is chosen using a shifted log-uniform distribution. This distribution is chosen to test degradation uniformly across multiple scales and include non-degraded tests; it is defined in the following formula using uniform number generator $u() \in [0, 1]$:

$$a \leftarrow \left\lfloor 2^{m \times u()} - 1 \right\rfloor$$

The chosen amount of equipment is then randomly removed from the complete topology. The resulting degraded (or complete) topology is routed at this point, and linear forwarding tables are dumped for analysis.

Evaluation of these tables is performed using static analysis of metrics representing maximum congestion risk for three communication patterns: all-to-all (A2A), random permutation (RP), shift permutation (SP). The congestion risk metric consists of counting $\min(\#srcs, \#dsts)$ for all routes of the corresponding pattern; this approximates network-caused congestion risk [15]. For A2A, the maximum congestion risk (throughout all ports) is the only value kept. RP consists of computing the maximum congestion risk for 1000 random permutations and keeping the median value. ($\sigma = 0{,}96$ for 100 RP samples in the case of Figure 2 with 256 randomly removed switches routed with Ftree.) SP consists of computing the maximum congestion risk for all $(\#N - 1)$ shift permutations and keeping the maximum value. (Shifts are based on the same node ordering which OpenSM's Ftree follows internally in order for quality comparison to be fair.) Such simplified performance models faithfully reflect comparative behaviour [16], though the absolute values measured are not good estimators of real throughput. Note that virtual channels potentially required by other algorithms are not taken into account in this analysis.

Congestion risk results are shown in Figure 2. When considering existing routing algorithms, Ftree provides the best performance for complete PGFTs (especially regarding SP for which the maximum congestion risk approaches theoretical optimal), but SSSP provides better stability under massive degradation, confirming results of the studies mentioned in Section 2. UPDN and MinHop provide visually identical results in this analysis: in fact in a full PGFT they are equivalent and vary only slightly under degradation. They both provide comparatively poor results for SP and A2A throughout the observed scale, however for RP they surprisingly improve significantly under massive degradation.

Dmodc provides minimal congestion risk throughout the considered range of degradations when compared with existing oblivious algorithms. In particular, it is even more stable than Ftree for SP under minimal degradation and nearly as stable as SSSP for A2A and RP under massive degradation.

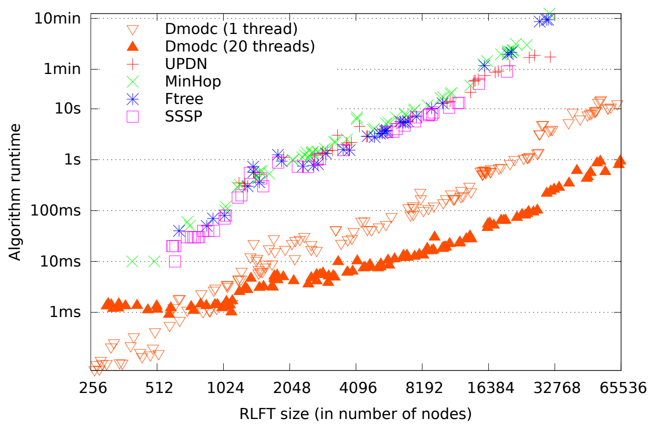

Figure 3. Algorithm runtime on an Intel Xeon E5-2680 v3 @ 2.50GHz (in log–log scale; lower is better).

Runtime

On our production implementation coded in C99, computation of cost, divider, topological NIDs, and routes are spread over POSIX threads fetching work with a switch-level granularity. Figure 3 reports complete algorithm execution time alongside OpenSM (version 3.3.21) routing times (measured by adding timers in the source code) running on the same machine. Note that local erraticness is partly due to our RLFT construction process for which the number of resulting switches is not monotonic with the number of requested nodes.

For clusters ranging up to many tens of thousands of nodes, Dmodc provides fast enough re-routing for a centralized fabric manager to react to faults before applications are interrupted.

5. Conclusion

The simulation results in Section 4 show that Dmodc provides high-quality centralized fault-resilient routing for PGFTs at a fraction of the runtime of existing algorithms, without relying on partial re-routing. Dmodc is also applicable to non-PGFT fat-tree-like topologies but with lower quality load balancing. As defined here, no effort has been made to minimize size of updates to be uploaded to switches throughout the fabric.

This algorithm is implemented inside BXI FM and has been succesfully deployed to an 8490 node PGFT production topology in which it helps provide fault-resiliency even when faced with thousands of simultaneous changes.

Acknowledgements

This research has been undertaken under a cooperation between CEA and Atos. with the goal of co-designing extreme computing solutions. Atos thanks CEA for all their inputs that were very valuable for this research. This research was partly funded by a grant of Programme des Investissements d'Avenir. This work has been jointly supported by the Spanish Ministry of Science, Innovation and Universities under the project RTI2018-098156-B-C52 and by JCCM under project SBPLY/17/180501/000498. BXI development was also part of ELCI, the French FSN (Fond pour la Société Numérique) cooperative project that associates academic and industrial partners to design and provide software components for new generations of HPC datacenters.

References

[1] E. Strohmaier, J. Dongarra, H. Simon, M. Meuer, and H. Meuer, "Top500 list," 1993–2018. [Online]. Available: www.top500.org

[2] C. Gómez, F. Gilabert, M. E. Gómez, P. López, and J. Duato, "Deterministic versus adaptive routing in fat-trees," in *International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2007, pp. 1–8.

[3] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, "Optimized InfiniBand™ fat-tree routing for shift all-to-all communication patterns," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 2, pp. 217–231, 2010.

[4] E. Zahavi, I. Keslassy, and A. Kolodny, "Quasi fat trees for hpc clouds and their fault-resilient closed-form routing," in *22nd Annual Symposium on High-Performance Interconnects (HOTI)*. IEEE, 2014, pp. 41–48.

[5] E. Zahavi, "D-mod-k routing providing non-blocking traffic for shift permutations on real life fat trees," *CCIT Report*, vol. 776, 2010.

[6] OpenSM, "Opensm – infiniband subnet manager," 2015. [Online]. Available: github.com/linux-rdma/opensm

[7] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho, "A survey and evaluation of topology-agnostic deterministic routing algorithms," *Transactions on Parallel and Distributed Systems (TPDS)*, vol. 23, no. 3, pp. 405–425, 2012.

[8] J. Domke, T. Hoefler, and S. Matsuoka, "Fail-in-place network design: interaction between topology, routing algorithm and failures," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Press, 2014, pp. 597–608.

[9] J.-N. Quintin and P. Vignéras, "Transitively deadlock-free routing algorithms," in *2nd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. IEEE, 2016, pp. 16–24.

[10] OpenSM, *Current OpenSM Routing § UPDN Routing Algorithm*, retrieved 2007. [Online]. Available: github.com/linux-rdma/opensm

[11] P. Vignéras and J.-N. Quintin, "Fault-tolerant routing for exascale supercomputer: The bxi routing architecture," in *International Conference on Cluster Computing (CLUSTER)*. IEEE, 2015, pp. 793–800.

[12] "Bull exascale interconnect." [Online]. Available: atos.net/en/products/high-performance-computing-hpc/bxi-bull-exascale-interconnect

[13] J. C. Villanueva, T. Skeie, and S.-A. Reinemo, "Routing and fault-tolerance capabilities of the fabriscale fm compared to opensm," Tech. rep. July, Tech. Rep., 2015.

[14] B. Bogdanski, B. D. Johnsen, S.-A. Reinemo, and F. O. Sem-Jacobsen, "Discovery and routing of degraded fat-trees," in *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*. IEEE, 2012, pp. 697–702.

[15] G. Rodriguez, R. Beivide, C. Minkenberg, J. Labarta, and M. Valero, "Exploring pattern-aware routing in generalized fat tree networks," in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 276–285.

[16] P. Faizian, M. A. Mollah, M. S. Rahman, X. Yuan, S. Pakin, and M. Lang, "Throughput models of interconnection networks: the good, the bad, and the ugly," in *25th Annual Symposium on High-Performance Interconnects (HOTI)*. IEEE, 2017, pp. 33–40.