Fail-in-Place Network Design: Interaction between Topology, Routing Algorithm and Failures

Jens Domke
Global Scientific Information
and Computing Center
Tokyo Institute of Technology, Japan
Email: domke.j.aa@m.titech.ac.jp

Torsten Hoefler
Computer Science Department
ETH Zurich, Switzerland
Email: htor@inf.ethz.ch

Satoshi Matsuoka
Global Scientific Information
and Computing Center
Tokyo Institute of Technology, Japan
Email: matsu@is.titech.ac.jp

***Abstract*—The growing system size of high performance computers results in a steady decrease of the mean time between failures. Exchanging network components often requires whole system downtime which increases the cost of failures. In this work, we study a fail-in-place strategy where broken network elements remain untouched. We show, that a fail-in-place strategy is feasible for todays networks and the degradation is manageable, and provide guidelines for the design. Our network failure simulation toolchain allows system designers to extrapolate the performance degradation based on expected failure rates, and it can be used to evaluate the current state of a system. In a case study of real-world HPC systems, we will analyze the performance degradation throughout the systems lifetime under the assumption that faulty network components are not repaired, which results in a recommendation to change the used routing algorithm to improve the network performance as well as the fail-in-place characteristic.**

***Keywords*—Network design, network simulations, network management, fail-in-place, routing protocols, fault tolerance, availability**

I. Introduction

Data centers or data warehouses as well as HPC systems grow to unprecedented scales. Tens of thousands of computers are integrated in large-scale facilities to process floods of queries or large compute jobs. The network connecting these machines plays a crucial role for successful operation. First, it has to guarantee a certain quality of service (bandwidth, latency) but also a certain availability (connectivity in case of failures). As network topologies grow, failures of switches and connectors or cables become common.

As opposed to network endpoints, the wiring complexity and infrastructural demands of cabling (e.g., arrangement of cable trays) make maintaining complex networks challenging and expensive. Thus, network structures are often kept in place over years and multiple generations of machines while other components such as CPU or main memory are upgraded. Today, many networks are based on the concept of over-provisioning the hardware, such as installing spare cables in the cable trays or having a spare parts inventory. Alternatively, they are operated in a deferred repair mode, in which a failed component will not be replaced instantaneously but within a reasonable timeframe, such as a business day.

Fail-in-place strategies are common in storage systems when maintenance costs exceed maintenance benefits, such as in large scale data centers with millions of hard drives. For example, Microsoft owned approximately one million servers in 2013, i.e., even an optimistic failure rate of 1% per year and two hypothetically hard drives per server would result in a mean time between failure of 26 minutes. Instead of replacing the hard drives of a server, the storage system, such as IBM's Flipstone [1], uses RAID arrays for reliability and a software approach to disable failed hard drives and to migrate the data, until a critical component failure disables the entire server.

In this paper, we define network fail-in-place based on the differentiation between "critical" and "non-critical" network component failures. A critical component failure disconnects all paths between two hosts, whereas a non-critical failure only disconnects a subset of all paths between two hosts. The network fail-in-place strategy is to repair critical failures only, but continue operation by bypassing non-critical failures. We explore in our practical study on artificial and real existing networks whether or not a fail-in-place strategy is a feasible approach for large-scale high performance and data center networks. This fail-in-place strategy is going to alter the future design process for HPC systems as well as the operation policies for the network.

First, we establish an empirical failure analysis, based on historic failures occurring in real systems, which will be used for subsequent failure and performance modeling. We proceed to show that it is not sufficient to consider fault properties (such as connectivity) of the topology or to consider the resiliency of the routing algorithm in isolation. Our analysis shows that, for InfiniBand networks, the choice of the routing algorithm influences the number of disconnected paths in case of a failure and the quality of service after a reinitialization (routing) of the topology with missing links or switches. Our main contributions and findings are the following:

- We show that fail-in-place network design can be accomplished with an appropriate combination of topology and routing algorithm while assuming a performance degradation up to a pre-defined threshold can be tolerated.
- Our toolchain allows system designers to plan future fail-in-place networks and operation policies while taking failure rates into consideration and allows administrators to evaluate the current state of the network by comparing it to the fault-free state.

SC14, November 16-21, 2014, New Orleans, Louisiana, USA
978-1-4799-5500-8/14/$31.00 ©2014 IEEE

TABLE I. COMPARISON OF NETWORK-RELATED HARDWARE AND SOFTWARE FAILURES, MTBF/MTTR, AND ANNUAL FAILURE RATES

Fault Type	Deimos[*]	LANL Cluster 2 [2]	TSUBAME2.0 [3]
Percentages of network-related failures			
Software	13%	8%	1%
Hardware	87%	46%	99%
Unspecified		46%	
Percentages for hardware only			
NIC/HCA	59%	78%	1%
Link	27%	7%	93%
Switch	14%	15%	6%
Mean time between failure / mean time to repair			
NIC/HCA	X[†] / 10 min	10.2 d / 36 min	X / 5–72 h
Link	X / 24–48 h	97.2 d / 57.6 min	X / 5–72 h
Switch	X / 24–48 h	41.8 d / 77.2 min	X / 5–72 h
Annual failure rate			
NIC/HCA	1%	X	≫ 1%
Link	0.2%	X	0.9%[‡]
Switch	1.5%	X	1%

- We conducted an exhaustive study of multiple InfiniBand (IB) routing algorithms (as implemented in OpenSM 3.1.16) and several real-world topologies. This allows us to draw conclusions about resilience properties of topologies and routing mechanisms, and to evaluate the applicability of their combinations for fail-in-place networks.
- For deterministically routed InfiniBand fat-trees, we show that two failing links may reduce the overall bandwidth by up to 30% and we observe rare curious cases where the performance of certain communication patterns increases with failing links.
- We conducted detailed case studies with two real HPC systems, TSUBAME2.0 and Deimos, and showed that the currently used routing method on TSUBAME2.0 can be improved, increasing the throughput by up to 2.1x (and 3.1x for Deimos) for the fault-free network while increasing their fail-in-place characteristics.

II. FAILURES IN REAL SYSTEMS

As other high performance computing hardware, networks are not immune to failures. The probability of network failures varies depending on the used interconnect hardware and software, system size, usage of the system, and age. Network failures constitute between 2%–10% for the HPC systems at Los Alamos National Laboratory [4], over 20% for LANs of Windows NT based computers [5] and up to 40% for internet services [6] of the total number of failures. Wilson et al. [7] showed a fairly constant failure rate of ≈7 unstable links between switches per month and a total of ≈30 disabled network interface controller (NIC) over the operation period of 18 month for a Blue Gene/P system.

The distribution of network component failures for three production systems shows that network-related software errors are less common compared to other hardware errors, but the actual distribution of switch failures, NIC/HCA failures and link failures varies heavily, see Tab. I. The first investigated HPC system, Deimos, in operation Mar.'07 – Apr.'12 at TU-Dresden, is a 728-node cluster with 108 IB switches and 1,653 links. The 49-node LANL Cluster 2 at Los Alamos National Laboratory was in operation Apr.'97 to Aug.'05. Installed in Sept.'10, TSUBAME2.0 at Tokyo Institute of Technology uses a dual-rail QDR IB network with 501 switches and 7,005 links to connect the 1,408 compute nodes via a full-bisection bandwidth fat-tree. Our simulations in Section V-D1 will be conducted based on the first rail, including all 1,555 nodes connected with 258 switches, and 3,621 links.

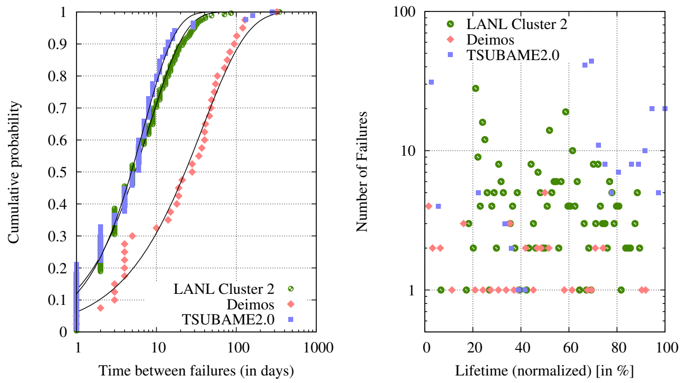
Fig. 1a shows the distribution of hardware faults over the life span of all three systems. Fitting the Weibull distribution to the data results in the following shape parameters for the different systems. For Deimos the shape parameter k is 0.76 and therefore the failure rate decreased over time [8]. The shape parameters for TSUBAME2.0, which is $k = 1.07$, and for the LANL Cluster 2, $k = 0.95$, indicate a constant failure rate. While the constant failure rate applies to the LANL cluster, it does not hold for TSUBAME2.0. The hardware faults of TSUBAME2.0, see Fig. 1b, show the expected bathtub curve for hardware reliability [9].

Common practice in high performance computing is to deactivate faulty hardware components and replace them during the next maintenance, unless the fault would threaten the integrity of the whole system. A failing switch can degrade the regular topology into an irregular topology which cannot be routed by a routing algorithm specifically designed for regular networks, even so the network is still physically connected. As shown in Tab. I the percentage of switch failures, among hardware-related failures of the network, ranges from 6% to 15% for the three systems. This covers issues of a faulty switch as well as faulty ports. One deactivated port of a switch will degrade a regular network topology, but replacing a whole switch for one broken port can be cost and time intensive.

For all network-related failures of the LANL Cluster 2 we calculate a mean time between failure (MTBF) of 100.3 h and a mean time to repair (MTTR) of 56.2 min. If we break this down into MTBF and MTTR for each network component, i.e., switch, link and network interface controller (NIC), we see that the MTTR is the smallest for NICs (36 min), followed by links (57.6 min) and switches (77.2 min). The MTBF for these components varies significantly. While every ten days one NIC experienced a failure, switches were four times as stable (MTBF=41.8 d). The most reliable network components of LANL Cluster 2 were the links. Only 23 link failures within eight years of operation were recorded, which results in a MTBF of more than three months. The system administrators of Deimos and TSUBAME2.0 followed a different approach of repairing faulty network components. While broken NICs (aka. HCAs in InfiniBand) of Deimos were replaced within 10 min by spare nodes, repairing links or switches took between 24 h and 48 h. Faulty components of TSUBAME2.0 were disabled within minutes, but for a switch port failure only the port and not the entire switch was disabled. Those disabled parts were replaced within the range of 5 h to one business day, i.e., up to 72 h time to repair. An exception were links and ports between root switches. Those were replaced in the next maintenance period, which was scheduled twice a year.

Table I summarizes failure percentages, MTBF/MTTR, and annual failure rates of all systems. The failure data of TSUBAME2.0 indicates an annual failure rate of ≈1% for InfiniBand links as well as 1% for the used switches. In the

[*] Deimos' failure data is not publicly available
[†] Not enough data for accurate calculation
[‡] Excludes first month, i.e., failures sorted out during acceptance testing



(a) CDF for faulty hardware

(b) Failures over time

Fig. 1. Network-related hardware failures

later sections, we will extrapolate the fail-in-place behavior of the network up to a system's lifetime of eight years based on this annual failure rate.

III. Building Blocks for Fail-in-Place Networks

Fail-in-place networks are only viable if the number of critical failures and resulting unavailability of the HPC system can be minimized. Therefore the following section will identify the needed hardware/software characteristics for the network.

A. Resilient Topologies

A high path-redundancy in the topology enhances the fail-in-place characteristic of an interconnection network. Theoretically, from the topology point of view, failures can be tolerated up to the point where a set of critical failures disconnects the network physically. Whether or not non-critical failures can disconnect the network through incomplete routing and their influence on the overall network throughput will be subject of the following sections.

The most commonly used topologies in high performance computing are torus networks as well as k-ary n-trees. Therefore, we evaluate the usability of 2D/3D meshes and tori [10], k-ary n-trees [11], and its more generalized version XGFTs [12], to build fail-in-place networks. Additionally, the emerging Dragonfly network topology [13], undirected Kautz graphs [14] and randomized topologies are investigated. Switches and links are used to build the base topology and the hosts are equally distributed over the switches for all investigated topologies but fat-trees. The hosts in a fat-tree are connected to the lowest level of the tree. For scalability and variability reasons both the intra-group network topology and the inter-group network topology of the Dragonfly topology were not predefined by Kim et al. [13]. In the following we are using 1D flattened butterfly topologies for the intra-group and inter-group topology.

1) Connectivity of the topologies: One resilience measurement of topologies is the connectivity of the graph $G(V, E)$ representing the network I [15]. The connectivity determines the minimum number of network components that have to be removed to disconnect the network, i.e., the worst case. However, neither does this estimate the average case nor can we derive throughput degradation from it. Let a path $p = (e_1, \ldots, e_n)$ from u to v be a sequence of edges $e_i \in E$ connecting a sequence of vertices $(u, v_1^*, \ldots, v_{j+n-1}^*, v)$. The (u, v)-vertex cut $\kappa(G; u, v)$ for each pair (u, v) of nodes in V is equal to the number of internally vertex-disjoint paths from u to v (for $(u, v) \notin E$). Similarly, the (u, v)-edge cut $\lambda(G; u, v)$ is equal to the number of edge-disjoint paths from u to v. The (vertex-)connectivity $\kappa(G)$ of a graph $G(V, E)$ is defined by the smallest number of (u, v)-vertex cuts

$$\kappa(G) = \min\{\kappa(G; u, v) : \forall u, v \in V \wedge (u, v) \notin E\} \qquad (1)$$

and its edge-connectivity $\lambda(G)$ is defined by

$$\lambda(G) = \min\{\lambda(G; u, v) : \forall u, v \in V\}. \qquad (2)$$

TABLE II. Vertex- and edge-connectivity for the investigated network topologies

Topology	Vertex-/Edge-Connectivity	Ref.
Mesh$(d_1, \ldots, d_n)$	n	[16]
Torus$(d_1, \ldots, d_n)$	$2 \cdot n$	[12]
k-ary n-tree	k	[12]
XGFT$(h, m_1, \ldots, \omega_1, \ldots)$	ω_1	[12]
Dragonfly(a, p, h, g) w/ 1D flattened butterfly	$a + h - 1$	[13]
Kautz(d, k) as undirected graph	$2 \cdot d$	[17]

A summary of the vertex-/edge-connectivity for the topologies listed above is presented in Tab. II. Unless mentioned otherwise, we consider terminal graphs $G(V, E, T)$ only, where $V \setminus T =: W$ represents the set of switches in I and T represents the set of terminal nodes (hosts) in I, and perform the connectivity analyses on the subgraph $G(W, E)$. Without edge-redundancy in the network the vertex- and edge-connectivity of I will be equivalent.

2) Redundancy to increase edge-connectivity: For a fair comparability of the different topologies in the following sections, we are using redundant links in the network to utilize all available switch ports. Hence, the topology is represented by an undirected multigraph. As a result, the number of hosts, switches and links can be balanced across the different investigated topologies. The link redundancy r, which increases the edge-connectivity but not the vertex-connectivity, is defined as the smallest number of parallel links between two adjacent switches

$$r := \min |\{(u, v) \in E : u, v \in W\}|. \qquad (3)$$

To simplify matters, we set r to one while analyzing the connectivity. For $r > 1$, the presented edge-connectivity in Tab. II needs to be multiplied by r.

B. Resilient Routing Algorithms

Two classes of deterministic routing algorithms are suitable to build fail-in-place interconnects: (1) fault-tolerant topology-aware routings that are able to compensate for minor faults within a regular topology and (2) topology-agnostic routing algorithms.

In terms of resiliency, topology-agnostic algorithms are superior to topology-aware routing algorithms when the network suffers from a high percentage of failures, as we will show in Section V-B. However, techniques to avoid deadlocks in topology-agnostic algorithms can limit their usage on large networks, such as the 3D torus(7,7,7) in Section V-C.

The InfiniBand subnet manger, called OpenSM, currently implements nine deterministic routing algorithms. Five of them, more precisely DOR routing [16], Up*/Down* and Down*/Up* routing [18], fat-tree routing [19] and Torus-2QoS routing [18], are representatives of the topology-aware algorithms. The remainder, namely MinHop [18] routing, DF-/SSSP routing [20], [21], and LASH routing [22], are topology-agnostic. While DOR falls into the category of topology-aware routing algorithms, its implementation in OpenSM can be considered topology-agnostic but it may not be deadlock-free for arbitrary topologies [16].

Deadlock-freedom of the routing algorithm is, besides latency, throughput and fault-tolerance, an essential property to design fail-in-place networks. Our simulations, described in Section V, reveal that even simple communication patterns such as all-to-all can cause deadlocks in the network. Once a deadlock is triggered the network either stalls globally or delays the execution if the interconnect possesses mechanisms for deadlock resolution.

Two other critical aspects of routing algorithms for fail-in-place networks are the runtime of the routing algorithm after a failure was discovered and the number of paths temporarily disconnected until the rerouting completed. Both problems depend on the routing algorithms and will be investigated hereafter.

C. Resiliency Metrics

To evaluate a fail-in-place network with its combination of topology and routing, common metrics such as bisection bandwidth cannot be applied directly. Important factors are availability and throughput despite the non-critical failures.

1) Disconnected paths before rerouting: Domke et al. [20] showed that the runtime of a routing step ranges from 10 ms for small networks to minutes for larger systems ($> 4,096$ HCAs). Until new LFTs are calculated a subset of the network might be disconnected. We are interested in the number paths that are disconnected by a link or switch failure.

First, we analyze the problem analytically using the edge-forwarding indices π_e [23] of the links in I, i.e., the number of routes passing through link e. Lets assume we have a routing algorithm $R : E \times T \mapsto E$ for a network $I := G(V, E, T)$, which maps the current edge e_i of a path towards $w \in T$ to the next edge e_{i+1} for all switches $v \in V$. Let R be perfectly balanced, i.e., the edge-forwarding index of I using this routing is minimal:

$$\pi(I) := \min_R \pi(I, R) = \min_R \max_{e \in E} \pi(I, R, e) \qquad (4)$$

see [23], [24] for further reference. Further, let π_e be defined as $\pi_e := \pi(I, R, e)$ for a fixed but arbitrary network I and perfectly balanced routing R. Removing an edge e_f from I will disconnect a path p, predetermined by R, iff $e_f \in p$. A disconnected path will not contribute to any edge- or vertex-forwarding index in I. We calculate the expected number of disconnected routes $\mathcal{E}(L)$ after n link failures, given by the set $L := \{e_i \in E : 1 \leq i \leq n \wedge e_i \neq e_j \text{ if } i \neq j\}$, for $n = 1$ with:

$$\mathcal{E}(\{e_1\}) = \frac{1}{|E|} \cdot \sum_{e \in E} \pi_e \qquad (5)$$

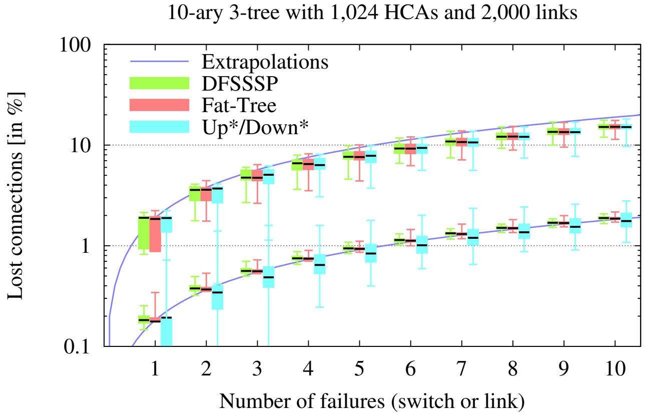


Fig. 2. Whisker plot for lost connections while removing switches (upper graphs) or links (lower graphs) without performing a rerouting

Unfortunately, for $n > 1$ we have to calculate $\mathcal{E}(L)$ as a conditional expected value, since some routes through e_f might have already been disconnected by removing another edge. Hence, $\mathcal{E}(L)$ cannot be calculated exactly without knowing the exact graph, routing algorithm and set L. Thus, we perform an empirical analysis for a 10-ary 3-tree topology. For each number of link and switch faults we created 100 topologies with randomly injected faults (seed $\in \{1, \ldots, 100\}$), and collected statistics of the number of lost connections. The result, see Fig. 2, shows that measured extremes depend on the used routing, but for a small number of faults and in a large network, the expected number of disconnected routes $\mathcal{E}(L)$ for $n > 1$ can be modeled by

$$\mathcal{E}(L = \{e_1, \ldots, e_n\}) \approx \frac{n}{|E|} \cdot \sum_{e \in E} \pi_e \qquad (6)$$

Knowing the runtime of the routing algorithm and the estimated temporary disconnected paths due to a network failure can be used to optimize the availability prediction of a system.

2) Throughput Degradation Measurement for Reliability Analysis: Evaluating a network can be accomplished with multiple techniques and metrics. System designers often use analytical metrics, such as bisection bandwidth, which is simple to analyze for some regular topologies, but NP-complete in the general case, e.g., when failures deform the network. During system operation, metrics based on actual benchmarks are accurate in terms of reflection of the real world, but require exclusive system access to remove the background traffic in the network. Therefore, a modeling technique is preferred, because it can be performed during the design phase of an HPC system and in parallel to the normal operation of the system, and parameters such as topology, failure, routing and communication pattern can be exchanged easily.

For the fail-in-place network design we rely on degradation modeling and the definition of a throughput threshold as a critical value. Modeling degradation over time in conjunction with a critical value, which defines the point of an unacceptable system state, are common in reliability analysis [25]. The degradation measurement with a simulator will show the communication throughput in the network during the increase of failed network components.

IV. Simulation Framework

A. Toolchain Overview

The first step in our toolchain for network reliability modeling is the generation of the network, including network faults, see Fig. 3. Either a regular topology, cf. Section III-A, is created or an existing topology file is loaded. The generator injects non-bisecting network component failures into the topology (users can also specify a fixed failure pattern reflecting an actual system state). Besides the actual network, the generator exports configuration files for the routing algorithms, such as the list of root switches for Up*/Down* routing.

The InfiniBand network simulator, called ibsim, in conjunction with OpenSM is used to configure the linear forwarding tables (LFTs) of all switches with the selected routing algorithm. In an intermediate step, the LFTs can be replaced to simulate a routing algorithm not included in OpenSM. The converter engine transforms the network and routing information into an OMNeT++ readable format.

A connectivity check is performed based on the LFTs. Incomplete LFTs are considered a failed routing, even if the routing within OpenSM did not report any problems while generating the LFTs. The last step is the network simulation with OMNeT++ [26] and the InfiniBand model [27].

OMNeT++ is one of the widely used discrete event simulation environments in the academic field. The InfiniBand model provides flit-level simulations of high detail and accuracy. In the following section, we briefly explain the model, and the extensions we made to the model, which we use to measure the throughput degradation for fail-in-place networks.

B. InfiniBand Model in OMNeT++

The simulated network itself consists of HCAs and switches and operates on the physical layer with a 8 bit/10 bit symbol encoding. The network components are configured for 4X QDR (32 Gbit/s data transfer rate), while the HCAs are plugged into 8X PCIe 2.0 5 GT/s slots. The PCI bus has a simulated hiccup every 0.1 μs which lasts for 0.01 μs to mirror the observed real world behavior. Our simulation environment further assumes that all links in the network are 7 m copper cables (longest passive copper cable for 40 Gbit/s QDR offered by Mellanox) and therefore produce a propagation delay of 43 ns for each flit, which is derived from the 6.1 ns/m delay mentioned in [28]. Switches are defined as a group of 36 ports connected via a crossbar which applies cut-through switching. Each switch input buffer can hold 128 flits per virtual lane, i.e., a total of 1,024 flits or 65,536 Bytes. Switch output buffers can only store 78 flits. Messages are divided into chunks of message transfer units – we chose to configure the MTU size to 2,048 byte – and the local route header (8 byte), base transport header (12 byte) and in/-variant CRC (4 byte / 2 byte) are added, which adds up to 2,074 byte for packets. The virtual lane arbitration unit is configured for fair-share among all virtual lanes. We verified all parameters with small networks, where we were able to calculate the expected network throughput and consumption rate at the sinks. Furthermore, we used a real testbed with 18 HCAs and one switch to benchmark the InfiniBand fabric and tune the simulation parameters accordingly.

C. Traffic Injection

We use unreliable connection (UC) for the transport mechanism to allow arbitrary message sizes and to omit the setup of queue pairs in the simulation. While using UC, the destination will not send an acknowledge message back to the source.

1) Uniform random injection: Uniform random injection should show the maximum throughput of the network which is measured at the sinks, i.e., we extract the consumption bandwidth at each sink after the simulation reached the steady state or the wall-time limit. A seeded Mersenne twister algorithm provides the randomness of destinations and repeatability across similar network simulations where the only difference is the used routing algorithm. The message size is 1 MTU.

2) Exchange pattern of varying shift distances: An algorithm determines the distances between all HCAs for the simulated network. Based on this distance matrix each HCA sends first to the closest neighbors with a shift of $s = 1$ and $s = -1$ and then in-/decrements the shift distance s up to $\pm\frac{|W|}{2}$ for the even case. For an odd number of HCAs, each HCA increments s up to $\pm\frac{|W|-1}{2}$ and subsequently performs the last shift of $s = \frac{|W|+1}{2}$. This exchange pattern is similar to the commonly used linear exchange pattern [29], except for the fact that it tries to optimize the traffic for full-duplex links. For the smaller simulation with ≈256 HCAs the message size is set to 10 MTU and for larger networks reduced to 1 MTU. The network throughput for the exchange pattern is calculated based on the number of HCAs, message size and runtime:

$$\textit{throughput} := \frac{\textit{\#HCA} \times (\textit{\#HCA} - 1) \times \textit{message size}}{\textit{runtime of exchange pattern}} \qquad (7)$$

D. Simulator Improvements

Simulations with the IB model run either for a configured time or until the user terminates it, even so only flow control messages are send between ports. Therefore, neither does the simulator detect the completion of the simulated traffic pattern nor does it detect deadlocks if the routing algorithm is not able to create deadlock-free routing tables for the topology.

1) Steady state simulation: Our flit-level simulator detects the steady state of a simulation via a bandwidth controller module, which is used for traffic patterns with an infinite number of flits, such as explained in Section IV-C1. If the average bandwidth (of all messages up to the "current" point in time) is within the a 99% confidence interval (CI), then the sink will send an event to the global bandwidth controller and report that the attached HCA is in a steady state. The global controller waits until at least 99% of all HCAs reported their steady state and then stops the simulation, assuming the network is in a steady state.

2) Send/receive controller: For simulations with finite traffic, such as the exchange pattern, the bandwidth controller is not applicable. Therefore, direct messages between the HCA generator/sink modules and a global send/receive controller are used. Each time the HCA creates a new message, it will determine the destination node, and send an event to the global controller. Sinks act similar and send an event each time the last flit of an IB message is received. The generators send a

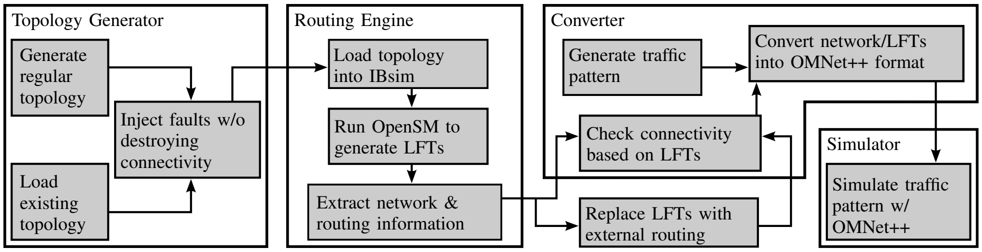

Topology Generator
Generate regular topology
Load existing topology
Inject faults w/o destroying connectivity
Routing Engine
Load topology into IBsim
Run OpenSM to generate LFTs
Extract network & routing information
Converter
Generate traffic pattern
Convert network/LFTs into OMNet++ format
Check connectivity based on LFTs
Replace LFTs with external routing
Simulator
Simulate traffic pattern w/ OMNet++

Fig. 3. Toolchain to evaluate the network throughput of a fail-in-place network

TABLE III. TOPOLOGY CONFIGURATIONS FOR A BALANCED (AND UNBALANCED) NUMBER OF HCAS W/ REDUNDANCY r

Topology	Switches	HCAs	Links	r
2D mesh(5,5)	25	275 (256)	240	6
3D mesh(3,3,3)	27	270 (256)	216	4
2D torus(5,5)	25	275 (256)	300	6
3D torus(3,3,3)	27	270 (256)	324	4
Kautz(2,4)	24	264 (256)	288	6
16-ary 2-tree	32	256 (270)	256	1
XGFT(1,22,11)	33	264 (256)	242	1
Dragonfly(10,5,5,4)	40	280 (256)	276	1
Random	32	256 (270)	256	1

TABLE IV. USABILITY OF TOPOLOGY/ROUTING COMBINATIONS; O : DEADLOCK-FREE; R : ROUTING FAILED; D : DEADLOCK DETECTED

	Fat-tree	Up*/Down*	DOR	Torus-2QoS	MinHop	SSSP	DFSSSP	LASH
	artificial topologies							
2D mesh	r	r	o	o	d	d	o	o
3D mesh	r	r	o	o	d	d	o	o
2D torus	r	r	d	o	d	d	o	o
3D torus	r	r	o	o	d	d	o	o
Kautz	r	r	d	r	d	d	o	o
k-ary n-tree	o	o	o	r	o	o	o	o
XGFT	o	o	o	r	o	o	o	o
Dragonfly	r	r	d	r	d	d	o	o
Random	r	r	o	r	d	d	o	o
	real-world HPC systems							
Deimos	r	o	o	r	o	o	o	o
TSUBAME2.0	o	o	o	r	o	o	o	o
	topology-aware				topology-agnostic			

second type of event when they have created the last message. The global send/receive controller keeps track of all scheduled messages and stops the simulation when the last flit of a traffic pattern arrives at its destination.

3) Deadlock controller: Deadlock detection is a complicated field and an accurate algorithm would require the tracking of actual flits and the available buffer spaces. Hence, our simulator uses a low-overhead distributed deadlock detection which is similar to the hierarchical deadlock detection protocol proposed by Ho and Ramamoorthy [30]. A local deadlock controller will observe the state of each port within a switch and reports periodically the state to a global deadlock controller. The three states of the switch are: idle, sending, and blocked (flits are waiting in the input buffers, but there is no free buffer space in the designated output port). The global deadlock controller will stop the simulation if and only if the state for each switch in the network is either blocked or idle.

V. SIMULATION RESULTS

A. Initial usability study

Not all routing algorithms work for all topologies. In the first part of our study, we characterize each routing algorithm for each topology regarding (1) if the algorithm is able to generate valid routes for the topology and (2) if these routes are deadlock-free. We list all used topologies (failure-free configuration) in Tab. III. We first investigate if the routing algorithm creates a valid set of routes. If it succeeds, then we use OMNeT++ to generate a random uniform injection load and check for deadlock situations. The result can be seen in Tab. IV. In the following we will omit all results for Down*/Up* routing since our tests have not shown any difference in performance compared to Up*/Down* routing.

As expected, all topology-aware routing algorithms are able to route their specific topologies. We note that dimension order routing (DOR) did not produce a deadlock on the 3x3x3 3D torus while the 5x5 2D torus configuration deadlocked. This is due to the one-hop paths on the 3x3x3 torus and we expect that larger configurations would produce deadlocks because DOR is only deadlock-free on meshes.

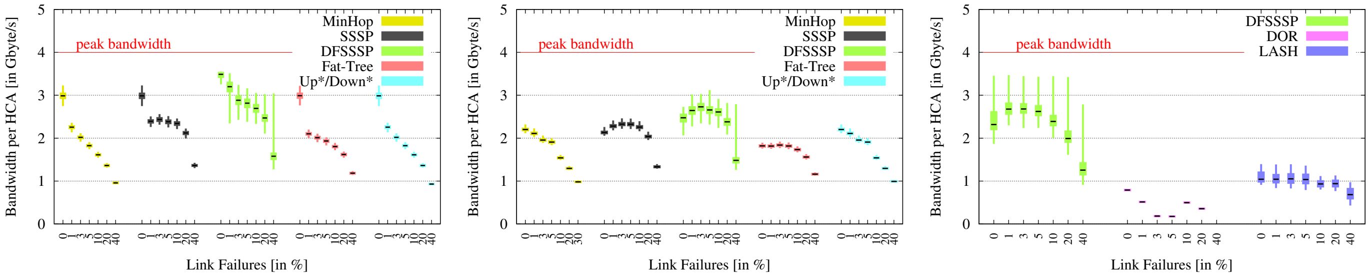
All topology-agnostic algorithms were able to route all topologies. However, MinHop and SSSP do not prevent deadlocks and thus create deadlocking routes in some configurations. We will proceed with the topology/routing pairs which are marked deadlock-free in Tab. IV in the following sections.

B. Influence of link faults on small topologies

We simulate uniform random injection and exchange communication patterns for all correct and deadlock-free routing algorithms. In the simulations, we determine the consumption bandwidth at the HCAs for the random uniform injection. The maximum injection and consumption bandwidth is 4 GByte/s, because 4X QDR is simulated. For the exchange pattern, we determine the time to complete the full communication by measuring the elapsed time until the last flit arrives at its destination, and we calculate the overall network throughput for this pattern according to Eq. 7.

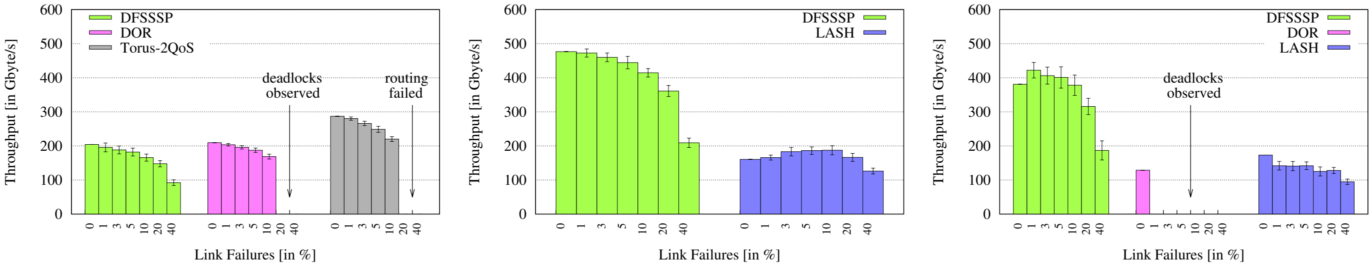
To simulate failure scenarios, we inject 1%, 3% and 5% random link failures (with three different seeds for uniform random injection and ten seeds for the exchange pattern; similar to the study conducted by Flich et al. [31]). In addition, we simulate more severe failure scenarios with 10%, 20% and 40% random link failures.

Two configurations, a balanced and an unbalanced, in terms of the number of HCAs per switch are simulated to see whether



(a) Balanced 16-ary 2-tree with 256 HCAs (b) Unbalanced 16-ary 2-tree with 270 HCAs (c) Random topology with 32 switches, 256 HCAs

Fig. 4. Whisker plots of consumption bandwidth for uniform random injection (box represents the three quartiles of the data, end of the whisker shows the minimum and maximum of the data; x-axis is not equidistant); Shown are the avg. values for three seeds (seed=1|2|3); Discussion in Section V-B



(a) Bal. 3D mesh$(3, 3, 3)$ with 270 HCAs, $r = 4$ (b) Bal. Dragonfly$(10, 5, 5, 4)$ with 280 HCAs (c) Random topology with 32 switches, 256 HCAs

Fig. 5. Histograms for exchange patterns with error bars (showing mean value and the 95% CI for all ten seeds (seed=$1, \ldots, 10$)); Missing bars: Deadlocks observed using DOR routing for at least one out of ten seeds, Torus-2QoS routing failed (multiple link failure in one 1D ring); Discussion in Section V-B

this has an influence on the maximal network throughput and resiliency or not, see Tab. III. The number of HCAs for the unbalanced configuration is listed in brackets in the third column of Tab. III. The balanced vs. unbalanced comparison is shown in Fig. 4a and Fig. 4b where we omit the plot of DOR and LASH routing because the resulting consumption bandwidth is less than 10% of DFSSSP's.

We analyzed the edge forwarding index for each simulation and we saw that LASH heavily oversubscribed certain links in the network while others are not used at all. Therefore, heavy congestion leads to the aforementioned consumption bandwidth decrease regardless of the link failure rate. To the best of our knowledge, the implementation of LASH in OpenSM optimizes for switch-to-switch traffic rather than terminal-to-terminal traffic which leads to poor load balancing on many of the investigated topologies.

We observe three general trends while comparing Fig. 4a and Fig. 4b: First, only 1% link failures—two faulty links for the 16-ary 2-tree topology—may result in a throughput degradation of 30% for uniform random injection on a fat-tree with full bisection bandwidth if the specialized fat-tree routing is used. With DFSSSP, for example, the degradation is only 8% for the median. Thus, we conclude that the choice of the routing method is important to maintain high bandwidths in the presence of network failures. Second, an imbalance in the number of HCAs per switch can have a similar influence on the throughput, even without faults in the network (≈40% degradation compared to the balanced fault-free case). While the resiliency of all routing algorithms is not affected by an unequal number of HCAs, which can be caused by compute node failures or caused by initial network design decisions, the network throughput will be affected. Third, we observe curious cases where the bandwidth increases with link failures. This indicates a suboptimal routing algorithm and we will explain this effect later.

Fig. 4c and Fig. 5c show the two communication patterns uniform random injection (Fig. 4c) and exchange (Fig. 5c) for the same random topology. We observe that random topologies are barely affected by link failures of less than 10%. We omit the results for the uniform random injection in the following analyses because low consumption bandwidth at the sinks perfectly correlates with a runtime increase of the exchange pattern, and therefore decrease in overall network throughput.

Torus performance is shown in Fig. 5a. The exchange pattern finishes 29% faster, i.e., 29% higher throughput, for Torus-2QoS compared to DFSSSP, but DFSSSP is more resilient, because Torus-2QoS was unable to create LFTs for some of the ten cases of 20% and 40% link failures, and the traffic pattern caused a deadlock using DOR routing for the mesh with 20% and 40% link failures.

Dragonfly topologies are usually routed with the Universal Globally-Adaptive Load-balanced routing schemes [13], which chooses between minimal path routing and Valiant's randomized non-minimal routing depending on the network load. The current architecture specification of InfiniBand uses deterministic routing, but for completeness we include simulations for the emerging Dragonfly topology in our study as well. Fig. 5b shows that both DFSSSP and LASH only degrade slowly with random network failures.

Statistical analysis of the data points for the exchange pattern for all investigated topologies shows a high correlation

TABLE V. Intercept [in Gbyte/s], slope, and R^2 for balanced topologies of Tab. III for best performing routing algorithm

Topology	Routing	Intercept	Slope	R^2
2D mesh	Torus-2QoS	263.63	-1.83	0.94
3D mesh	Torus-2QoS	276.18	-2.37	0.87
2D torus	Torus-2QoS	341.11	-1.68	0.95
3D torus	DFSSSP	508.97	-2.12	0.90
Kautz	DFSSSP	299.48	-1.45	0.88
16-ary 2-tree	DFSSSP	629.76	-3.59	0.69
XGFT	DFSSSP	527.31	-3.03	0.88
Dragonfly	DFSSSP	479.03	-2.39	0.94
Random	DFSSSP	417.53	-2.17	0.76

between the number of failed links in the topology and the network throughput of the exchange pattern. We analyze the coefficient of determination (R^2), assuming a linear model, for every combination of topology and routing algorithm.

We propose the following method as part of the design process for a new fail-in-place network: The best routing algorithm, among the investigated algorithms, for a fail-in-place network will be chosen based on the intercept and slope of the linear regression. The intercept approximates the overall network throughput of the exchange pattern for the fault-free network configuration whereas the slope predicts the throughput decrease (in Gbyte/s) for each non-critical hardware failure which is kept in-place. Therefore, even hybrid approaches are possible where the system administrator switches between routing algorithms after a certain number of failures is reached, if the regression line of the routing algorithms intercept at this point.

Intercept and slope are the main indicators for the quality of the routing whereas the coefficient of determination is an important metric how well the throughput can be explained by the number of failures. The results for the coefficient of determination calculation for our investigated topology/routing combinations are summarized in Tab. V. This shows that the performance of the exchange pattern on small topologies can be explained almost entirely by one variable, namely link failures. We list the best performing routing algorithm based on intercept and slope for each topology in Tab. V.

C. Influence of link faults on large topologies

For the larger network sizes we reduce the percentage of link failures. Assuming a constant annual failure rate of 1%, derived from the analyzed failure rates of TSUBAME2.0 in Section II, we simulate $1\%, \ldots, 8\%$ links failures which will give an estimate of the performance degradation of the network for the system's lifetime.

We identified three key points while investigating the larger topologies: First, the investigated fault-tolerant topology-aware routing algorithms are resilient for realistic link failure percentages, but their fail-in-place characteristic in terms of network throughput is worse than topology-agnostic algorithms. The comparison of DFSSSP routing and fat-tree routing in Fig. 6b illustrates the issue.

The second fact is that not all topology-agnostic routing algorithms are applicable to every topology above a certain size. Fig. 6c shows the comparison of throughput degradation between four balanced large-scale topologies with ≈2,200 HCAs. DFSSSP routing is the best performing algorithm for the Dragonfly topology and the 14-ary 3-tree, but failed to create LFTs for the torus and the Kautz graph due to limitations of available virtual lanes. The same issue causes LASH routing to be unusable on the 7x7x7 torus.

TABLE VI. Intercept [in Gbyte/s], slope, and R^2 for the large topology/routing combination shown in Fig. 6c

Topology	#HCAs	Routing	Intercept	Slope	R^2
3D torus(7,7,7)	2,058	Torus-2QoS	2,794.89	-2.06	0.66
Dragonfly(14,7,7,23)	2,254	DFSSSP	2,166.54	-1.50	0.34
Kautz(7,3)	2,352	LASH	1,541.58	0.09	0.02
14-ary 3-tree	2,156	DFSSSP	5,015.53	-1.85	0.76

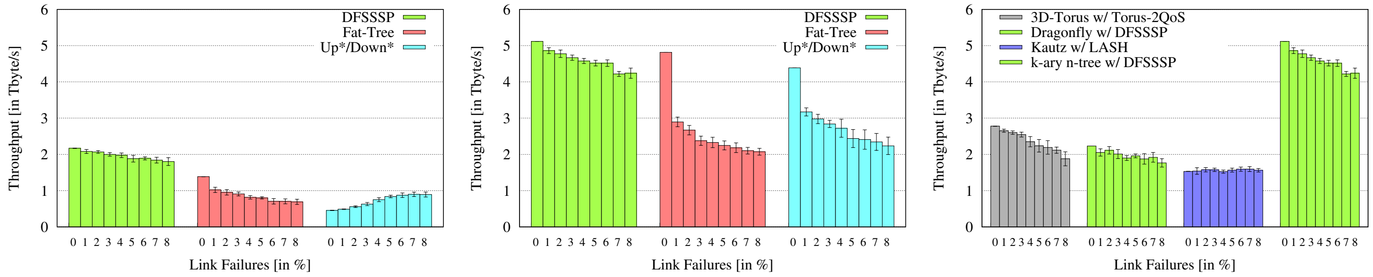
And third, the low coefficient of determination R^2, see Tab. VI, indicates that the number of link failures is not the dominating influence on the runtime of the exchange pattern on larger topologies. Hence, a low R^2 for the best of the investigated routing algorithms means that the throughput does not decline heavily during the system's lifetime which is a desired characteristic for a fail-in-place network. We presume the match or mismatch between the configured routes and the communication pattern to be a second explanatory variable, which will be explained further based on Fig. 6a.

In Fig. 6a, we also identified an anomaly: A major increase in network throughput using Up*/Down* routing is possible while the number of failed links in the network increases. This effect has two reasons, first: A comparison with DFSSSP and fat-tree routing, which enable higher throughput, indicates a serious mismatch between communication pattern and static routing algorithm in the fault-free case, which has been shown by Hoefler et al. [32]. The second reason is that each failure in the fabric will change the deterministic routing, which can lead to an improvement in runtime for the same communication pattern. This change in the routing can remove congestion, because simultaneously executed point-to-point communications share less links/switches in the fabric.

D. Case studies of TSUBAME2.0 and Deimos

For these case studies we will combine randomized link failures with switch failures, i.e., a form of localized simultaneous failures of many links, to mirror the behavior of a real-world HPC system.

1) Tsubame2.0: Based on the fault data presented in Section II, we calculated an annual link failure rate of 0.9% and an annual switch failure rate of 1% for TSUBAME2.0. The same data indicates a failure ratio of $1 : 13$, i.e., thirteen switch-to-switch links will fail for each failing switch. In the following, we assume a constant annual failure rate of 1% for the links and switches and we analyze the performance degradation if only links will fail, see Fig. 7a, and if only switches will fail, shown in Fig. 7b. This results in the simulation of eight years of TSUBAME2.0's lifetime (the anticipated lifetime of the network used in TSUBAME2.0 and TSUBAME2.5 is six years). In addition, we use the failure ratio of $1 : 13$, to mirror the real-world behavior, to investigate the network performance and resiliency of the routing algorithms for TSUBAME2.0, as shown in Fig. 7c. In all figures we omit the results for Down*/Up*, since they are equal to Up*/Down* routing, and we omit the results for LASH, because of a two orders of



(a) 10-ary 3-tree with 1,100 HCAs (b) 14-ary 3-tree with 2,156 HCAs (c) Different networks with ≈2,200 HCAs

Fig. 6. Histograms (mean value and 95% CI; ten runs w/ seeds $\in \{1, \ldots, 10\}$) for exchange patterns for networks with up to 8% link failures; Figure (c) compares 3D torus(7,7,7)|2,058 HCAs, Dragonfly(14,7,7,23)|2,254, Kautz(7,3)|2,352, 14-ary 3-tree|2,156 (from left to right); Discussion in Section V-C

TABLE VII. Intercept, slope, and R^2 for TSUBAME2.0 and Deimos shown in Fig. 7c and Fig. 8c (default: italic; best: bold)

HPC system	Routing	Intercept [in Gbyte/s]	Slope	R^2
TSUBAME2.0	**DFSSSP**	1,393.40	-1.33	0.62
	Fat-Tree	1,187.19	-1.48	0.66
	Up/Down**	717.76	-0.08	0.01
Deimos	*MinHop*	29.94	-	-
	DFSSSP	93.40	-0.15	0.09
	Up*/Down*	30.10	0.06	0.11

magnitude lower throughput, see Tab VII. The default routing algorithm on TSUBAME2.0 is Up*/Down*, but fat-tree routing performs better for this exchange pattern, as we see in Fig. 7a. DFSSSP routing outperforms both.

Statistical analysis of the data presented in Fig. 7c shows that the intercept for fat-tree routing is 15% lower compared to DFSSSP routing while the slope of the linear regression line is 10% steeper compared to DFSSSP, see Tab VII. In conclusion, the change of the routing algorithm to DFSSSP on TSUBAME2.0 would not only lead to a higher performance of the exchange pattern on the fault-free network, but also will increase the fail-in-place characteristic of the network.

2) Deimos: The annual failure rate is 0.2% for links and 1.5% for switches, as listed in Tab. I. The switch-to-link fault ratio can be approximated with $1 : 2$. We performed the case study of Deimos similarly to the study of TSUBAME2.0. For each number of link and/or switch failures, we simulated ten different faulty network configuration, while injecting randomized failures with seeds $\in \{1, \ldots, 10\}$.

The conclusion derived from the results shown in Fig. 8 and Tab. VII is that a change from the default routing algorithm (MinHop) to DFSSSP routing would have increased the throughput for exchange patterns by a factor of three. Remarkable is the fact that a fail-in-place approach for Deimos would have had almost no effect on the performance of the communication layer, meaning that maintenance costs for the network—except for critical failures—could have been saved completely.

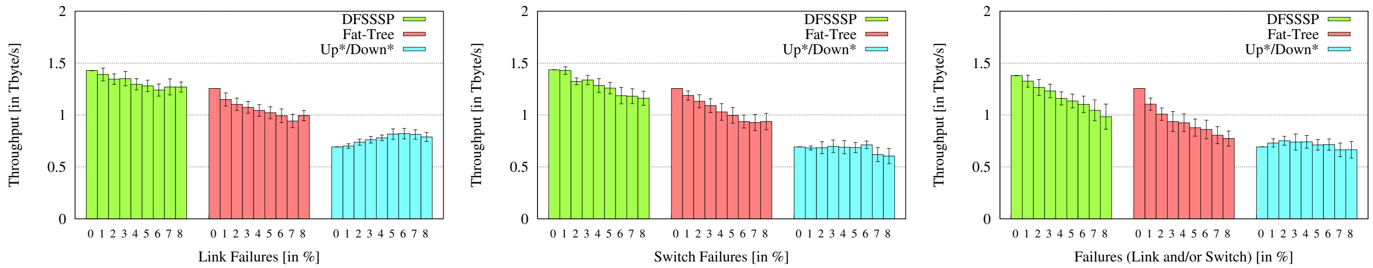
VI. Related Work

Resiliency of network topologies has been studied extensively in the past. Xu et al. [15] analyzed numerous interconnection topologies, such as De Bruijn and Kautz graphs, and mesh and butterfly networks. However, an analysis of routing algorithms has been done based on edge forwarding index, but not based on actual performance delivered for a traffic pattern. Another comprehensive survey of network fault tolerance was performed by Adams et al. [33]. The authors describe methods to increase fault tolerance through hardware modification of existing topologies, such as adding additional links, ports or switches. Other attempts to enhance the fault tolerance properties of the network have been performed analytically. One example is the multi-switch fault-tolerant modified four tree network which aims to improve the single-switch fault-tolerant four tree network [34]. Another example is the attempt to solve the problem heuristically with genetic algorithms [35].

Co-design strategies have been proposed, such as the F10 network [36], where the interconnection topology is optimized for the routing algorithm and vice versa. This includes an increase in alternative paths using an AB Fat-Tree instead of a ordinary fat-tree and includes enhanced fault recovery properties through local rerouting. The routing algorithm for F10, besides optimizing for global load balancing, manages the failure detection and propagates this information to a subset of the nearest neighbors of the failure. This subset is smaller for F10 than in a fat-tree allowing for fast recovery and therefore decreases the packet loss.

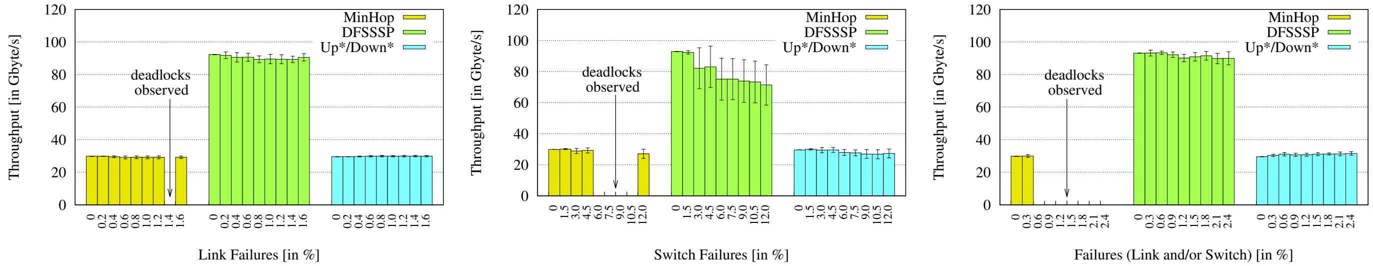
Okorafor et al. [37] proposed an fault-tolerant routing algorithm for an optical interconnect, which assumes a fail-in-place 3D mesh topology and enhances the throughput. The IBM Intelligent Bricks project [38] investigates the feasibility of building fail-in-place storage systems and includes considerations about the performance loss in the network due to failed bricks. The bandwidth degradation in the 6x6x6 3D torus is simulated up to 60% failed bricks. However, no other routing algorithm besides the IceCube mesh routing has been tested.

Fault-tolerant topology-aware routing algorithms have been developed and studied in the past, especially for meshes and tori topologies and in the field of Network-on-Chip, e.g., [39]–[43], but their usability for other topologies has not been tested. Flich et al. [31] performed a survey of topology-agnostic routing algorithms and compared them not only analytically, but also evaluated their performance with a flit-level simulator. The simulation has been conducted on small 2D mesh and tori networks, up to a size of 16x8. Additionally to these regular topologies, faulty meshes and tori with an injected number of link failures of 1%, 3%, and 5% were investigated.



(a) Link failures only (1% annual failure rate) (b) Switch failures only (1% annual failure rate) (c) Switch and link failures (1 : 13 ratio)

Fig. 7. Histograms for exchange patterns for different failure types using DFSSSP, fat-tree and Up*/Down* routing on TSUBAME2.0 (LASH excluded because throughput is only 2% of DFSSSP's); Shown: mean value and 95% CI for ten runs per failure percentage (seeds $\in \{1, \ldots, 10\}$); Discussion in Section V-D1



(a) Link failures only (0.2% annual failure rate) (b) Switch failures only (1.5% annual failure rate) (c) Switch and link failures (1 : 2 ratio)

Fig. 8. Histograms for exchange patterns for different failure types using MinHop, DFSSSP and Up*/Down* routing on Deimos (LASH excluded because throughput is only 9% of DFSSSP's); Shown: mean value and 95% CI for ten runs per failure percentage (seeds $\in \{1, \ldots, 10\}$); Deadlocks observed using MinHop routing for at least one out of ten seeds; Discussion in Section V-D2

VII. Implications for HPC system design and operation policies

We have shown that HPC networks, and not only hard drives, can be operated in fail-in-place mode to reduce maintenance costs and to utilize the remaining resources. Even so the resulting irregular topologies pose a challenge to the used routing algorithms, a low failure rate of the network components supports a fail-in-place network design strategy, which bypasses non-critical failures during the system's lifetime.

System designer can readily use our toolchain to make decisions about the correct combination of topology and available routing algorithm. Estimated failure rates will help to extrapolate the performance degradation of the fail-in-place network and will help to derive operation policies from those results. During operation, our toolchain allows system administrators to quantify the "current" state (degradation) of the system due to failures. Depending on the predefined degradation threshold the administrator then can decide whether or not a maintenance has to be scheduled. The complete toolchain can be downloaded from our webpage http://spcl.inf.ethz.ch/Research/Scalable_Networking/FIP and allows researchers, system designer, and administrators to reproduce our work and conduct studies with other systems.

VIII. Conclusion

As we have seen in Section V, to guarantee high overall network throughput over the system's lifetime the resilient topology has to be combined with an appropriate fault-tolerant or topology-agnostic routing algorithm. Both types of deterministic routing algorithms, fault-tolerant topology-aware and topology-agnostic, show limitations. The performance degradation using a topology-aware routing algorithm increases more with an increase of failures compared to topology-agnostic routings, and a large number of switch and link failures can deform a regular topology to a state which cannot be compensated by the topology-aware routing, because it cannot recognize the underlying topology. Even so topology-agnostic routing algorithms are supposed to be ideal for fail-in-place networks, the investigated algorithms either show weaknesses in terms of deadlocks, such as MinHop, or cannot be used beyond a certain network size, because the deadlock-avoidance via virtual lanes would require more than the available number of virtual lanes, e.g., in the case of LASH and DFSSSP for the 3D torus(7,7,7) with $> 2,000$ HCAs.

Our analysis of lost connections after a failure, and time before a rerouting step is finished, is valuable for other fault tolerance areas as well. E.g., the knowledge about the runtime of a rerouting step can be used to calculate appropriate retry counters and timeout values in the communication layer. The information about routing paths, the average network failure rate and resulting path disconnects can improve existing scheduling strategies to reduce the number of applications which share the same link/switch. Hence, the number of potential application crashes caused by a single fault will be reduced.

This paper provides a comprehensive empirical study of combinations of topologies, routing algorithms and network failures. Besides the usage for fail-in-place network design, our developed simulation environment can be used in multiple fields of research and development of new networks and routing algorithms. Not only can we derive from that the resilience of the topologies or the resilience of the routing, but we also can make recommendation about the hardware and software configuration of existing InfiniBand networks, such as the suggested change of the routing algorithm for TSUBAME2.0 (and the already decommissioned Deimos).

ACKNOWLEDGMENT

The authors would like to thank Eitan Zahavi for making the initial IB module for OMNeT++ publicly accessible, and the researchers at Simula Research Laboratory for their effort to port the original IB module to the newest OMNeT++ version. Last but not least, the authors would like to thank the HPC system administrators at LANL, Technische Universität Dresden and Tokyo Institute of Technology for collecting highly detailed failure data of their systems and sharing the information.

REFERENCES

[1] M. Banikazemi, J. Hafner, W. Belluomini, K. Rao, D. Poff, and B. Abali, "Flipstone: Managing Storage with Fail-in-place and Deferred Maintenance Service Models," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 1, pp. 54–62, Jan. 2008.

[2] Los Alamos National Laboratory, "Operational Data to Support and Enable Computer Science Research," Apr. 2014, https://institute.lanl.gov/data/fdata/.

[3] Global Scientific Information and Computing Center, "Failure History of TSUBAME2.0 and TSUBAME2.5," Apr. 2014, http://mon.g.gsic.titech.ac.jp/trouble-list/index.htm.

[4] B. Schroeder and G. A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," in *Proceedings of the International Conference on Dependable Systems and Networks*, ser. DSN '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 249–258.

[5] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer, "Failure Data Analysis of a LAN of Windows NT Based Computers," in *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, ser. SRDS '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 178–187.

[6] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, ser. USITS'03. Berkeley, CA, USA: USENIX Association, 2003, pp. 1–15.

[7] L. J. Wilson, "Managing vendor relations: a case study of two HPC network issues," in *Proceedings of the 24th international conference on Large installation system administration*, ser. LISA'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–13.

[8] W. Nelson, *Applied Life Data Analysis*, ser. Wiley Series in Probability and Statistics. Wiley, 2004.

[9] A. Verma, A. Srividya, and D. Karanki, *Reliability and Safety Engineering*, ser. Springer series in reliability engineering. Springer, 2010.

[10] G. Wang and J. Chen, "On fault tolerance of 3-dimensional mesh networks," in *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks*, May 2004, pp. 149–154.

[11] 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*, Apr. 1997, pp. 87–93.

[12] S. R. Öhring, M. Ibel, S. K. Das, and M. J. Kumar, "On generalized fat trees," in *IPPS '95: Proceedings of the 9th International Symposium on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 1995, p. 37.

[13] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 77–88.

[14] D. Li, X. Lu, and J. Su, "Graph-Theoretic Analysis of Kautz Topology and DHT Schemes," in *NPC*, 2004, pp. 308–315.

[15] J. Xu, *Topological Structure and Analysis of Interconnection Networks*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[16] T. Rauber and G. Rünger, *Parallel Programming: for Multicore and Cluster Systems*. Springer, 2013.

[17] L. Hsu and C. Lin, *Graph Theory and Interconnection Networks*. Taylor & Francis, 2008.

[18] *Mellanox OFED for Linux User Manual*, Rev 2.0-3.0.0 ed., Mellanox Technologies, Sunnyvale, CA, USA, Oct. 2013.

[19] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, "Optimized InfiniBand fat-tree routing for shift all-to-all communication patterns," *Concurr. Comput. : Pract. Exper.*, vol. 22, no. 2, pp. 217–231, Feb. 2010.

[20] J. Domke, T. Hoefler, and W. E. Nagel, "Deadlock-Free Oblivious Routing for Arbitrary Topologies," in *Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*. Washington, DC, USA: IEEE Computer Society, May 2011, pp. 613–624.

[21] T. Hoefler, T. Schneider, and A. Lumsdaine, "Optimized Routing for Large-Scale InfiniBand Networks," in *17th Annual IEEE Symposium on High Performance Interconnects (HOTI 2009)*, Aug. 2009.

[22] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, 1987.

[23] F. R. K. Chung, E. Coffman, Jr., M. Reiman, and B. Simon, "The forwarding index of communication networks," *IEEE Trans. Inf. Theor.*, vol. 33, no. 2, pp. 224–232, Mar. 1987.

[24] G. Hahn and G. Sabidussi, *Graph Symmetry: Algebraic Methods and Applications*, ser. NATO ASI series / C: NATO ASI series. Springer, 1997.

[25] C. Croarkin, P. Tobias, J. J. Filliben, B. Hembree, W. Guthrie, L. Trutna, and J. Prins, Eds., *NIST/SEMATECH e-Handbook of Statistical Methods*. NIST/SEMATECH, Jan. 2014, http://www.itl.nist.gov/div898/handbook/.

[26] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–10.

[27] E. G. Gran and S.-A. Reinemo, "InfiniBand congestion control: modelling and validation," in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '11. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 390–397.

[28] D. Blake and W. Gore, "Effect of Passive and Active Copper Cable Interconnects on Latency of Infiniband DDR compliant systems," Sep. 2007, http://www.openfabrics.org/archives/2007infiniband/10_W.L.Gore&Associates.pdf.

[29] B. Prisacari, G. Rodriguez, C. Minkenberg, and T. Hoefler, "Bandwidth-optimal all-to-all exchanges in fat tree networks," in *Proceedings of the 27th international ACM conference on International conference on supercomputing*, ser. ICS '13. New York, NY, USA: ACM, 2013, pp. 139–148.

[30] G. Ho and C. Ramamoorthy, "Protocols for Deadlock Detection in Distributed Database Systems," *IEEE Transactions on Software Engineering*, vol. SE-8, no. 6, pp. 554–557, 1982.

[31] 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," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 405–425, Mar. 2012.

[32] T. Hoefler, T. Schneider, and A. Lumsdaine, "Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks," in *Proceedings of the 2008 IEEE International Conference on Cluster Computing*. IEEE Computer Society, Oct. 2008.

[33] I. Adams, G.B., D. Agrawal, and H. Siegel, "A Survey and Comparision of Fault-Tolerant Multistage Interconnection Networks," *Computer*, vol. 20, no. 6, pp. 14–27, 1987.

[34] S. Sharma and P. K. Bansal, "A new fault tolerant multistage interconnection network," in *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 1, 2002, pp. 347–350 vol.1.

[35] E. Szlachcic, "Fault Tolerant Topological Design for Computer Networks," in *International Conference on Dependability of Computer Systems. DepCos-RELCOMEX '06*, 2006, pp. 150–159.

[36] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A Fault-tolerant Engineered Network," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 399–412.

[37] E. Okorafor and M. Lu, “Percolation routing in a three-dimensional multicomputer network topology using optical interconnection,” *J. Opt. Netw.*, vol. 4, no. 3, pp. 157–175, Mar 2005.

[38] C. Fleiner, R. Garner, J. Hafner, K. K. Rao, D. Kenchammana-Hosekote, W. Wilcke, and J. Glider, “Reliability of modular mesh-connected intelligent storage brick systems,” *IBM Journal of Research and Development*, vol. 50, no. 2.3, pp. 199–208, 2006.

[39] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[40] Y.-J. Suh, B. V. Dao, J. Duato, and S. Yalamanchili, “Software based fault-tolerant oblivious routing in pipelined networks,” in *Proc. of the 1995 International Conference on Parallel Processing. August 1995, I 101 - I*, 1995, p. 105.

[41] A. Ben Ahmed and A. Ben Abdallah, “Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3D-network-on-chip (3D-NoC),” *The Journal of Supercomputing*, vol. 66, no. 3, pp. 1507–1532, 2013.

[42] K. L. Man, K. Yedluri, H. K. Kapoor, C.-U. Lei, E. G. Lim, and J. M, “Highly Resilient Minimal Path Routing Algorithm for Fault Tolerant Network-on-Chips,” *Procedia Engineering*, vol. 15, no. 0, pp. 3406–3410, 2011.

[43] A. Rezazadeh, M. Fathy, and G. Rahnavard, “Evaluating Throughput of a Wormhole-Switched Routing Algorithm in NoC with Faults,” in *International Conference on Network and Service Security, 2009. N2S ’09.*, 2009, pp. 1–5.