Fail-in-Place Network Design

Interaction between Topology, Routing Algorithm and Failures

Jens Domke[♯], Torsten Hoefler[♮], Satoshi Matsuoka[♯]

[♯] Tokyo Institute of Technology
[♮] ETH Zürich

Presentation Overview

1. Topologies, Routing, Failures

2. Resilience Metrics

3. Simulation Framework

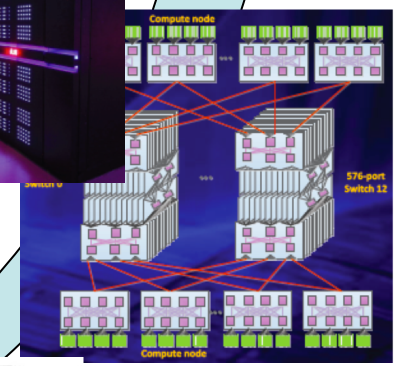
4. Influence of Failures

5. Lessons Learned & Conclusions

HPC Systems / Networks

Massive networks needed to connect all compute nodes of supercomputer!

2013: Tianhe-2 (NUDT)
16,000 Nodes
Fat-Tree



2011: K (RIKEN)
82,944 Nodes
6D Tofu Network

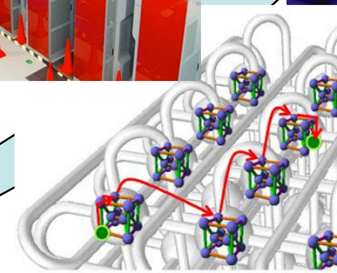
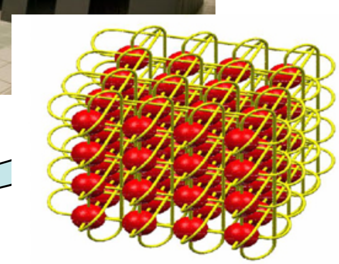
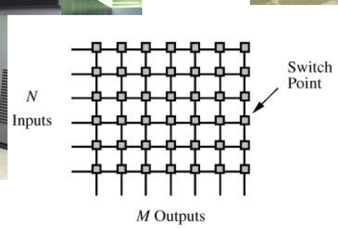


Fig. 6 TOFU Routing Algorithms

2004: BG/L (LLNL)
16,384 Nodes
3D-Torus Network

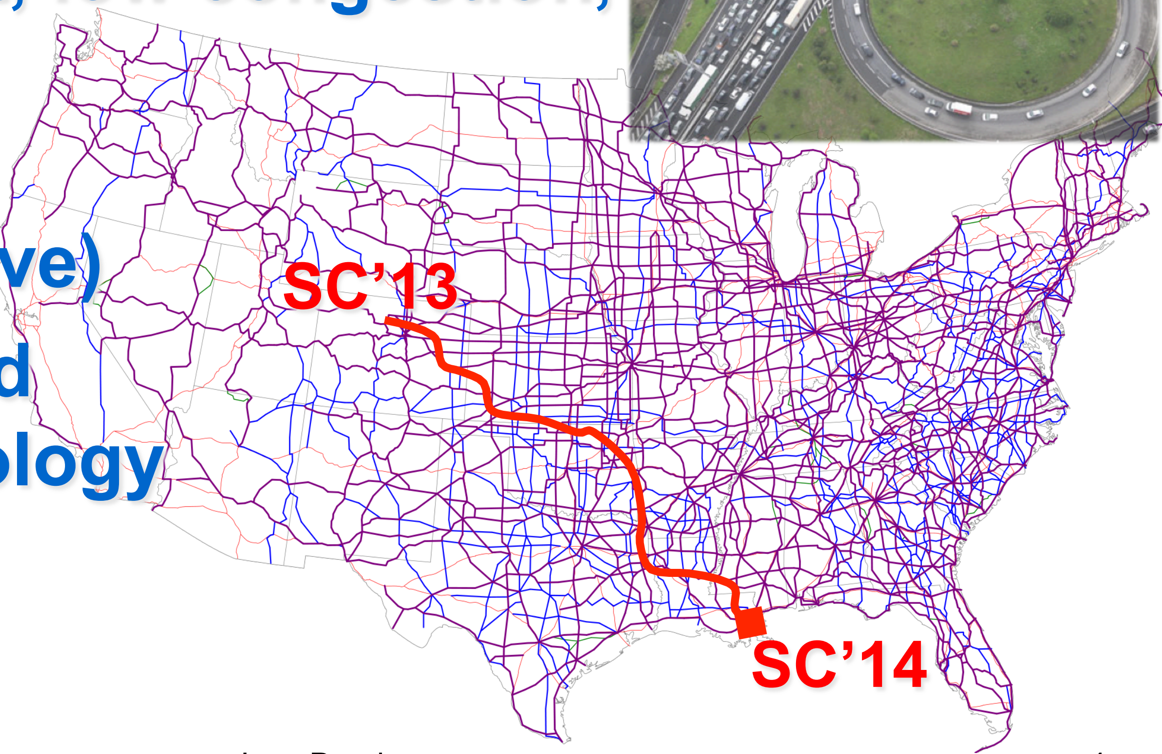


1993: NWT (NAL)
140 Nodes
Crossbar Network



Routing in HPC Network

- Similarities to car traffic, …
- Key requirements: low latency, high throughput, low congestion, fault-tolerant, deadlock-free
- Static (or adaptive)
- Highly depended on network topology and technology



Routing Algo. Categories

Topology-aware

- ☺ Highest throughput
- ☺ Fast calculation of routing tables
- ☺ Deadlock-avoidance based on topology characteristics
- ☹ Designed only for specific type of topology
- ☹ Limited fault-tolerance

Topology-agnostic

- ☺ Can be applied to every connected network
- ☺ Fully fault-tolerant
- ☹ Throughput depends on algorithm/topology
- ☹ Slow calculation of routing tables
- ☹ Complex deadlock-avoidance (CDG/VLs or prohibited turns)

[Flich, 2011]

Failure Analysis

- **LANL Cluster 2 (97–05)**
  - Unknown size/config.
- **Deimos (07–12)**
  - 728 nodes; 108 IB switches; ≈1,600 links
- **TSUBAME2.0/2.5 (10–?)**
  - 1,555 nodes (1,408 compute nodes); ≈500 IB switches; ≈7,000 links
- **Software more reliable**
- **High MTTR**
- **≈1% annual failure rate**
- **Repair/maintenance is expensive!**

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

Fault Type	Deimos*	LANL Cluster 2	TSUBAME2.5
	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%

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

Fail-in-Place Strategies

- Common in storage systems
- Example: IBM's Flipstone [Banikazemi, 2008] (uses RAID arrays; software disables failed HDD, migrates data)
- Replace only *critical* failures, and disable *non-critical* failed components
- Usually applied when maintenance costs exceed maintenance benefits

Can we do the same in HPC networks?

Presentation Overview

1. Topologies, Routing, Failures

2. **Resilience Metrics**

3. Simulation Framework

4. Influence of Failures

5. Lessons Learned & Conclusions

Network Metrics

- **Extensively studied in literature, but ignores routing**
  - E.g., (bisection) bandwidth, latency, diameter, degree
    ⤷ NP-complete for arbitrary/faulty networks
- **Topology resilience alone is not important**
- **Network connectivity doesn't ensure routing connectivity (especially for topology-aware algorithms)**

We need different metrics for fail-in-place networks!

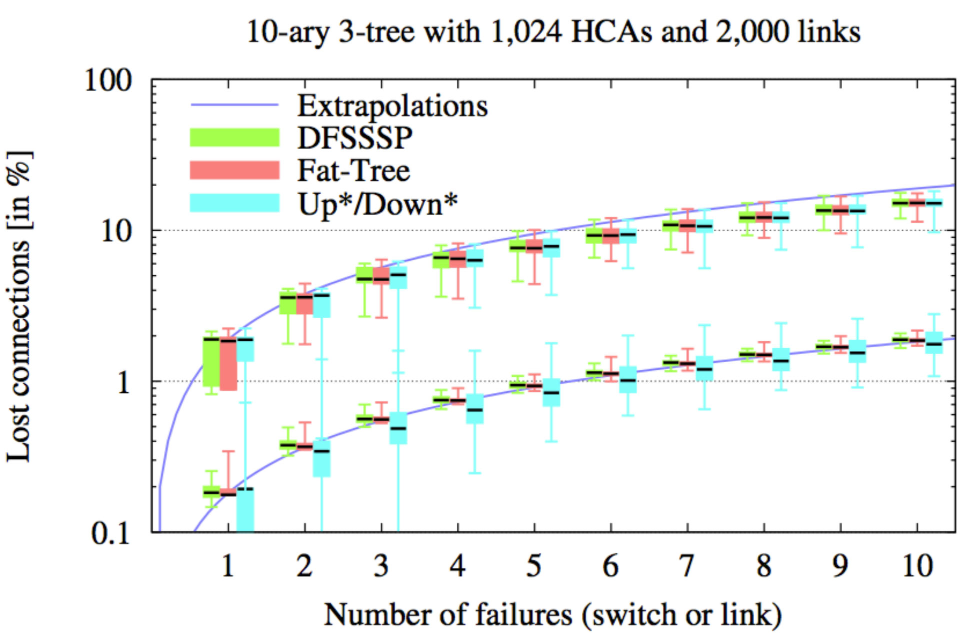
Disconnected Paths

- Important for availability estimation and timeout configuration for MPI, IB, …
- Rerouting can take minutes [Domke, 2011]
- For small error counts it can be extrapolated by

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

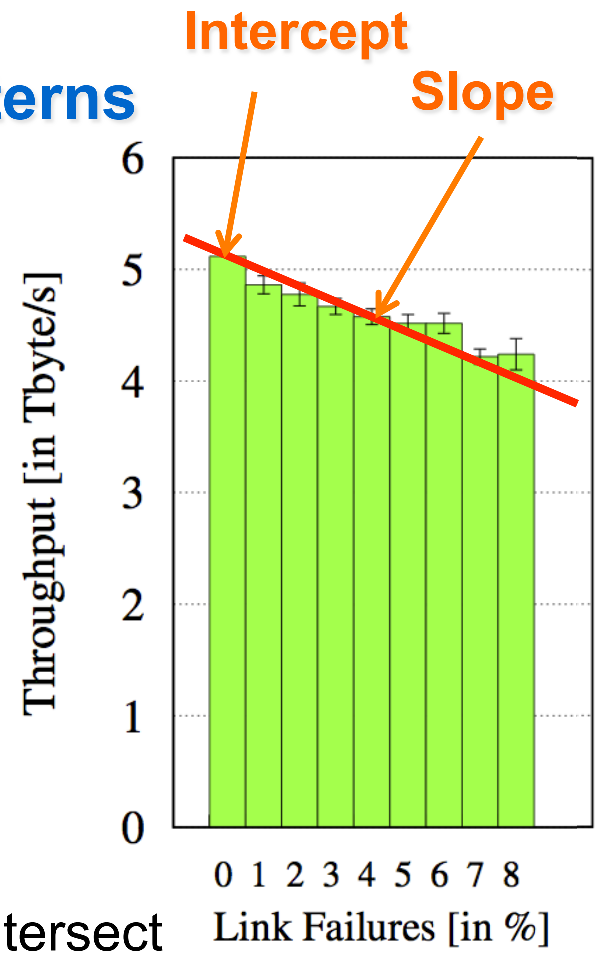
i.e., multiples of the avg. edge forwarding index π_e

- 100 random fault injections for each error count ➔



Throughput Degradation

- **Fault-dependent degradation measurement for fixed traffic patterns**
- **Multiple random faulty networks per failure percentage (seeded)**
- **Linear regression to gather intercept, slope, R^2 coeff. of determination**
- **Good routing: high intercept, slope close to 0, R^2 close to 1**
- **Possible conclusions**
  - Compare quality of routing algorithms
  - Change routing if two lin. regressions intersect



Presentation Overview

1. Topologies, Routing, Failures

2. Resilience Metrics

3. **Simulation Framework**

4. Influence of Failures

5. Lessons Learned & Conclusions

IB Flit-level Simulation

- **OMNet++ 4.2.2**
  - Discrete event simulation environment
  - Widely used in academia and open-source
- **IBmodel for OMNet++ [Gran, 2011]**
  - InfiniBand model developed by Mellanox
  - 4X QDR IB (32Gb/s peak); 7m copper cables (43ns propagation delay); 36-port switches (cut-through switching); max. 8 VLs; 2,048 byte MTU, flit = 64 byte
  - Transport: unreliable connection (UC) → no ACK msg
  - Tuned all simulation parameters with a real testbed with 1 switch and 18 HCAs

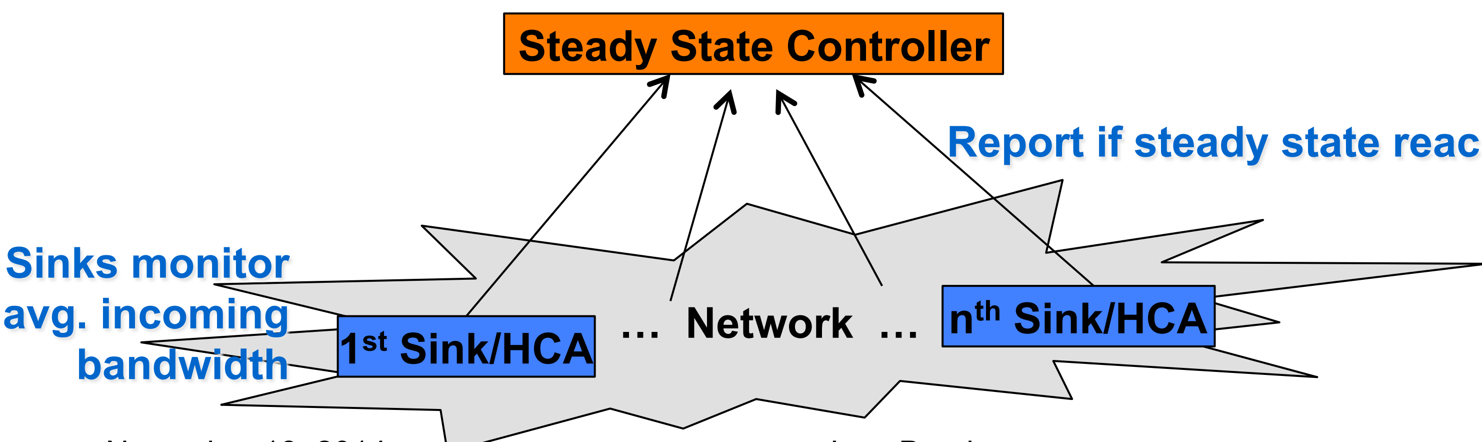
Traffic Injection

- **Uniform random injection**
  - Infinite traffic generation (message size: 1 MTU)
  - Show the max. network throughput (measure at sinks)
  - Seeded Mersenne twister for randomness/repeatability
- **Exchange pattern of varying shift distances**
  - Finite traffic (message size: 1 or 10 MTU)
  - Determine distances between all HCAs
  - Send first to closest neighbors (w/ shift s=±1)
  - In-/decrements the shift distance up to $\pm \frac{|\#HCA|}{2}$

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

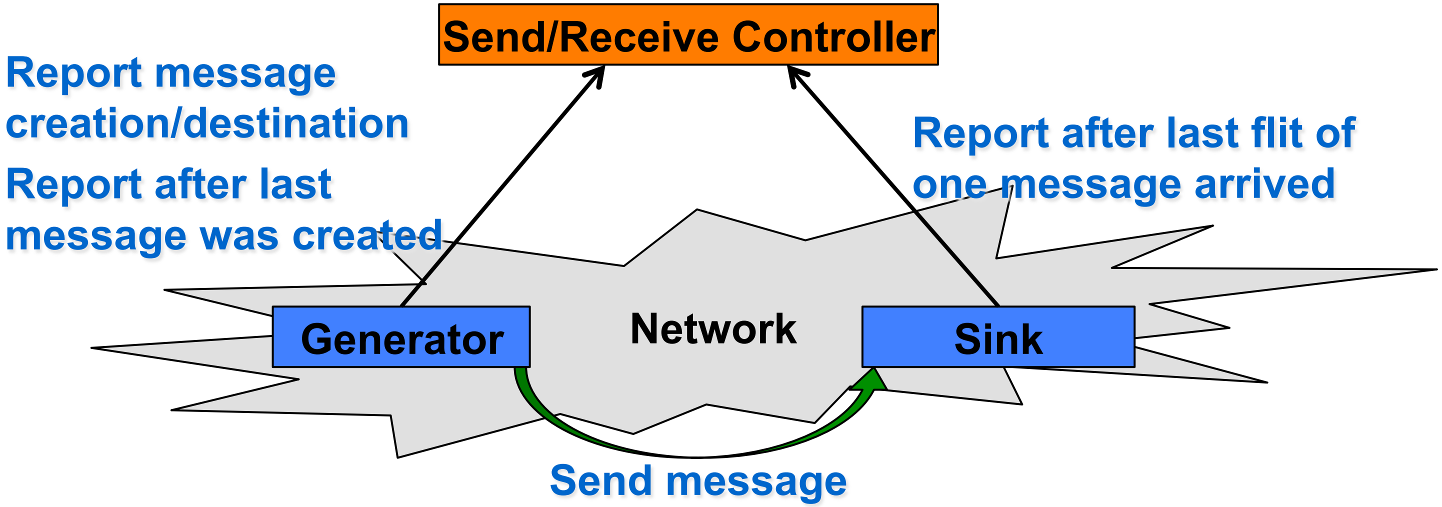
Enhancements

- **Default OMNet++ behaviour**
  - Runs for configured time or until termination by user
  - Flow control packets in IBmodel ➔ no termination
- **Steady state simulation (for uniform random)**
  - Stop simulation if sink bandwidth is within a 99% confidence interval for at least 99% of the HCAs



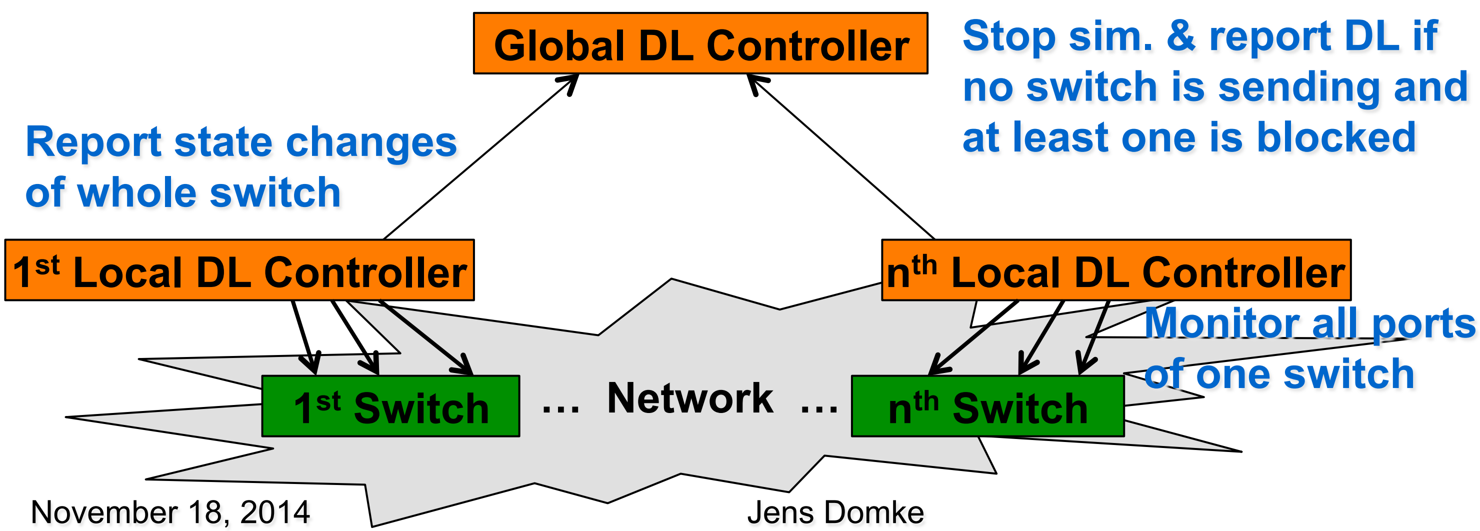
Enhancements

- **Send/receive controller (for exchange traffic)**
  - Steady state controller not applicable
  - Generator/sink modules (of HCAs) report to global send/receive controller
  - Controller stops simulation after last message arrived



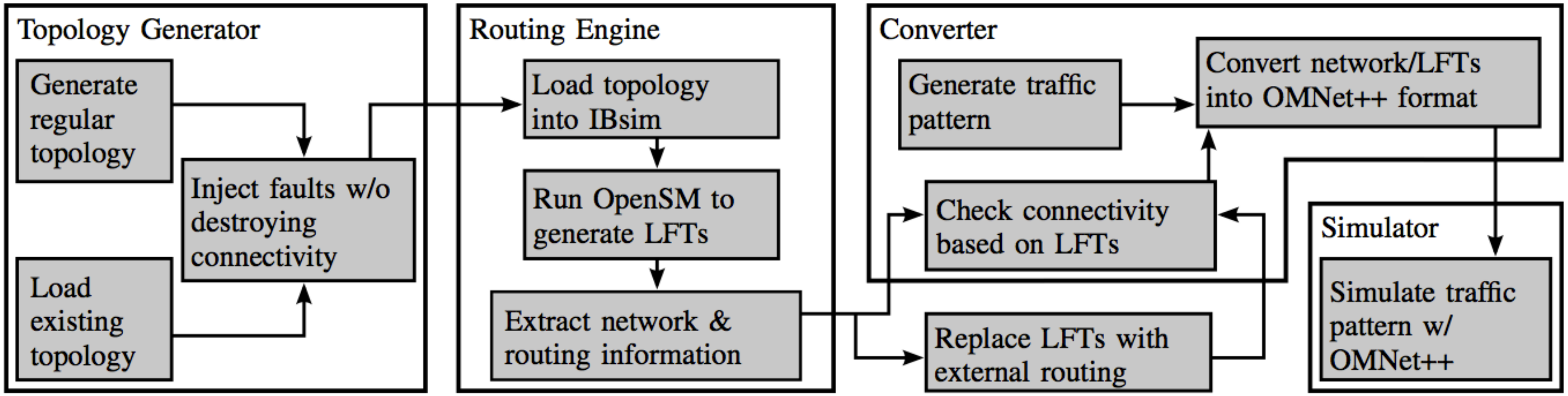
Enhancements

- **Deadlock (DL) controller**
  - Accurate DL detection too complex (runtime)
  - Low-overhead distributed DL-detection based on hierarchical DL-detection protocol [Ho, 1982]
  - Local DL controller observes switch ports (states: idle, sending, and blocked); reports to global DL controller;



Simulation Toolchain

- Generate faulty topology based on artificial/real network (preserve physical connectivity)
- Apply topology-[aware | agnostic] routing & check logical connectivity
- Convert to OMNet++ readable format
- Execute [random | all-2-all] traffic simulation



Presentation Overview

1. Topologies, Routing, Failures

2. Resilience Metrics

3. Simulation Framework

4. **Influence of Failures**

5. Lessons Learned & Conclusions

Valid Combinations

Use toolchain to try all in OpenSM implemented routing algorithms with all topologies (small artificial and real HPC)

DOR imple. in OpenSM is not really topology-aware

➔ deadlocks for some networks

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

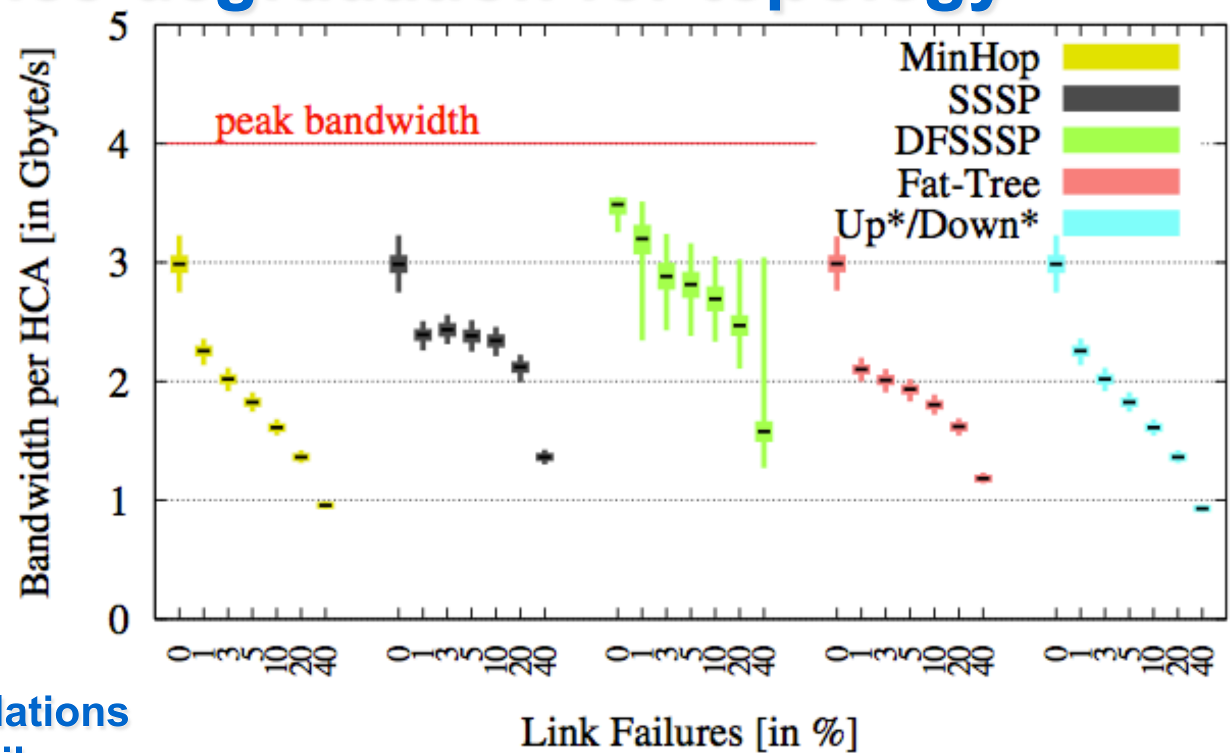
	topology-aware				topology-agnostic			
	Fat-tree	Up*/Down*	DOR	Torus-2QoS	MinHop	SSSP	DFSSSP	LASH
	artificial topologies							
2D mesh	r	r	o	o	d	d	o	o
3D mesh	r	r	o	o	d	d	o	o
2D torus	r	r	d	o	d	d	o	o
3D torus	r	r	o	o	d	d	o	o
Kautz	r	r	d	r	d	d	o	o
k-ary n-tree	o	o	o	r	o	o	o	o
XGFT	o	o	o	r	o	o	o	o
Dragonfly	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

Small Failure = Big Loss

1% link failures (= two faulty links) results in 30% performance degradation for topology-aware routing algorithms

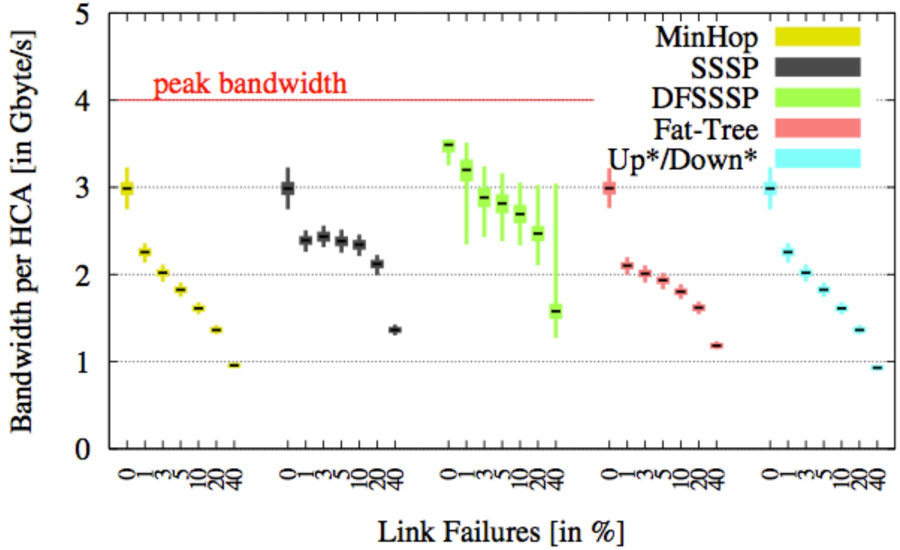
- Whisker plots of consumption BW at sinks
- VL usage results in DFSSSP's fan out

(avg. values from 3 simulations with seeds=[1|2|3] per failure percentage)



Balanced 16-ary 2-tree with 256 HCAs

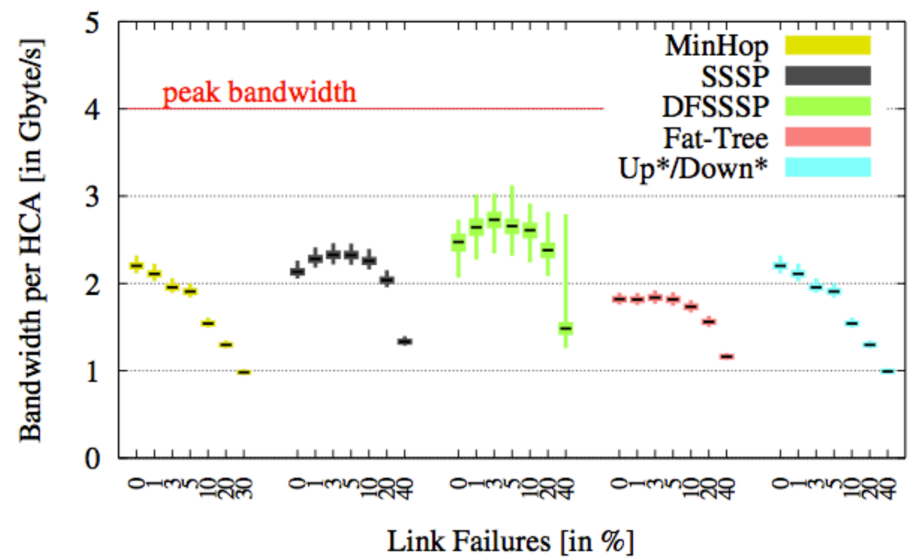
Balanced vs Unbalanced



Balanced 16-ary 2-tree with 256 HCAs

1% link failures (= two faulty links) can yield up to 30% performance degradation

Unbalanced network configuration (i.e., unequal #HCA/switch) can have same effect



Unbalanced 16-ary 2-tree with 270 HCAs

Topo.-aware vs agnostic

For some topologies neither topology-aware nor topology-agnostic routing algorithms perform well.

Topology-agnostic

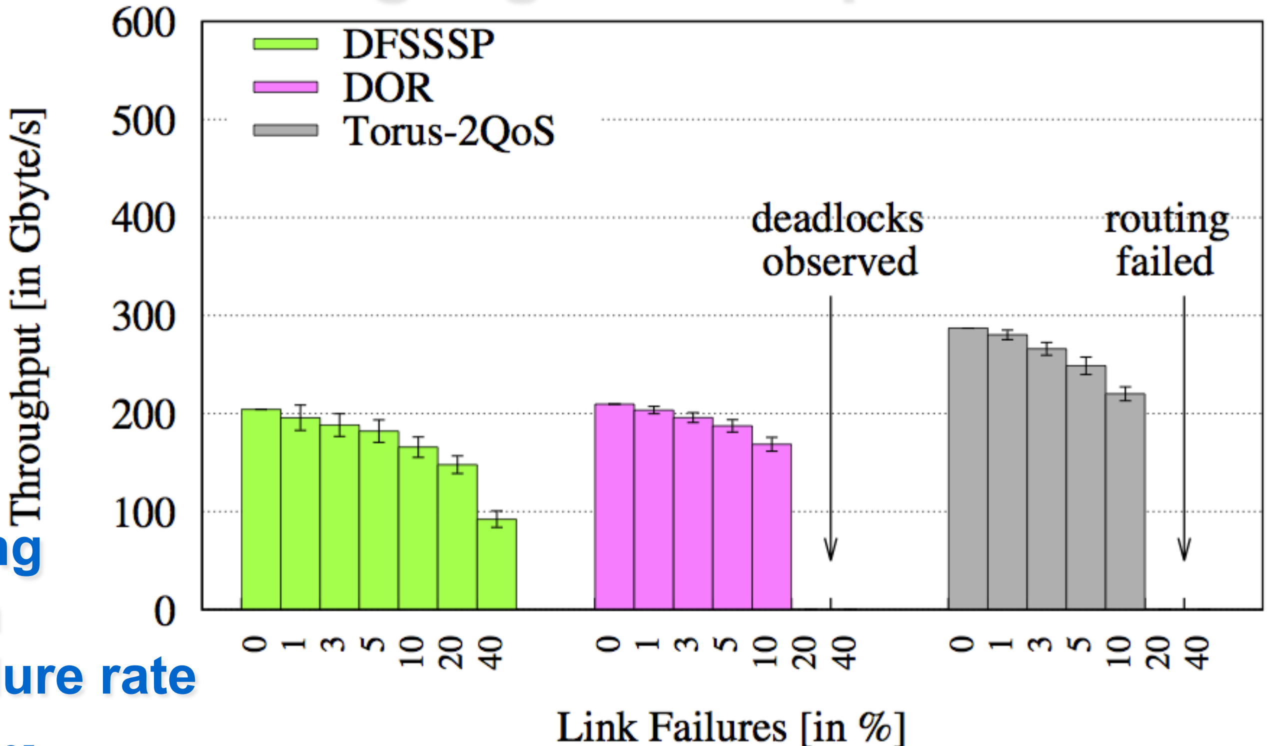
- Low throughput

Topology-aware

- Not resilient enough

➔ Solution: changing routing algorithm depending on failure rate

(10 sim. with seeds=[1..10] per failure percentage)

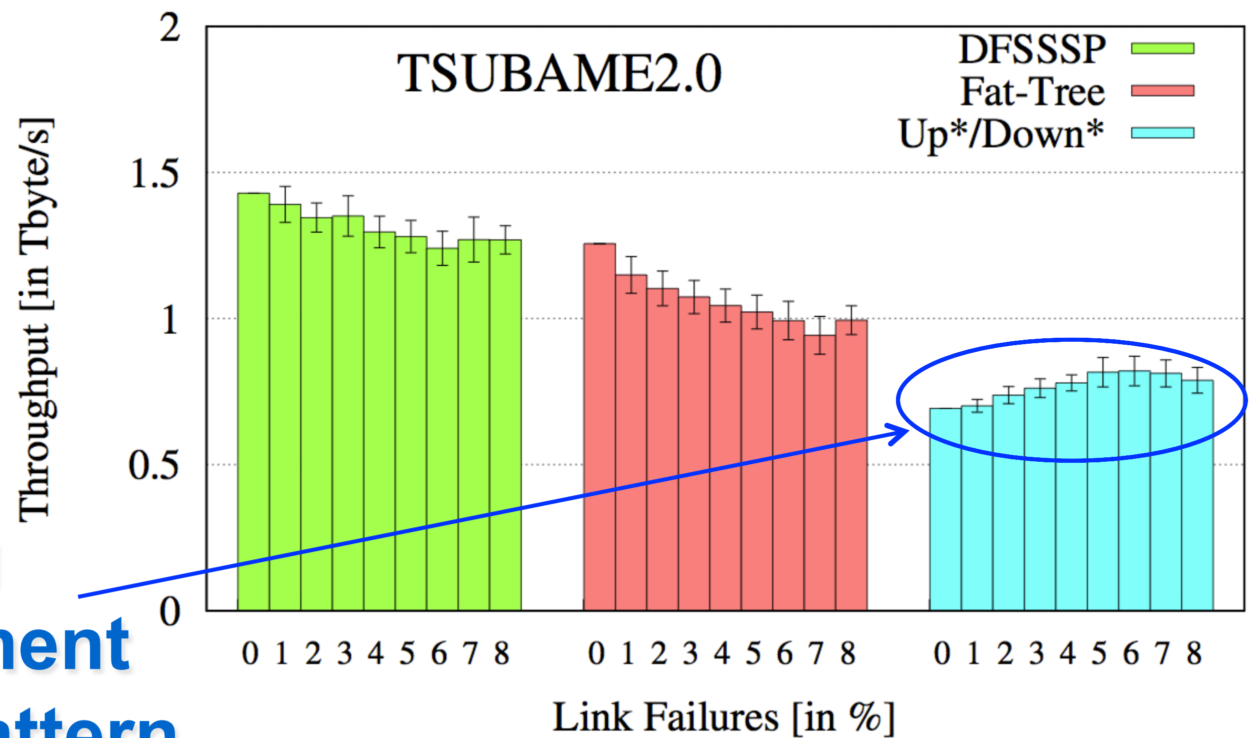


Bal. 3D mesh$(3,3,3)$ with 270 HCAs, $r = 4$

Failure↗ ≟ Throughput↗

Serious mismatch between static routing and traffic pattern results in low throughput for the fault-free case [Hoefler, 2008]

Failures will change the deterministic routing leading to an improvement for the same pattern



Link failures only (1% annual failure rate)

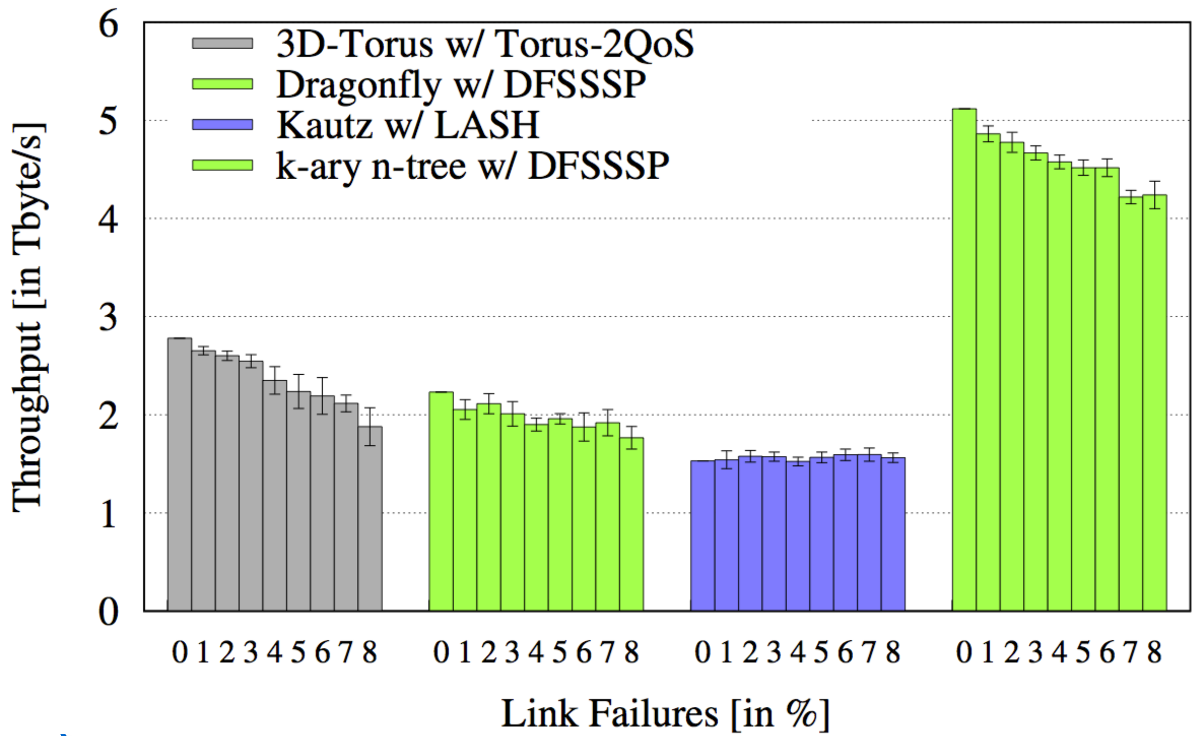
Routing at Larger Scales

- **DFSSSP & LASH failed to route the 3D torus**
- **Kautz graph either very resilient or bad routing**

Working routing

- **3D torus**
  - Torus-2QoS
- **Dragonfly**
  - DFSSSP, LASH
- **Kautz graph**
  - LASH
- **14-ary 3-tree**
  - DFSSSP, LASH
    Fat-Tree, Up*/Down*

(Only best routing shown)

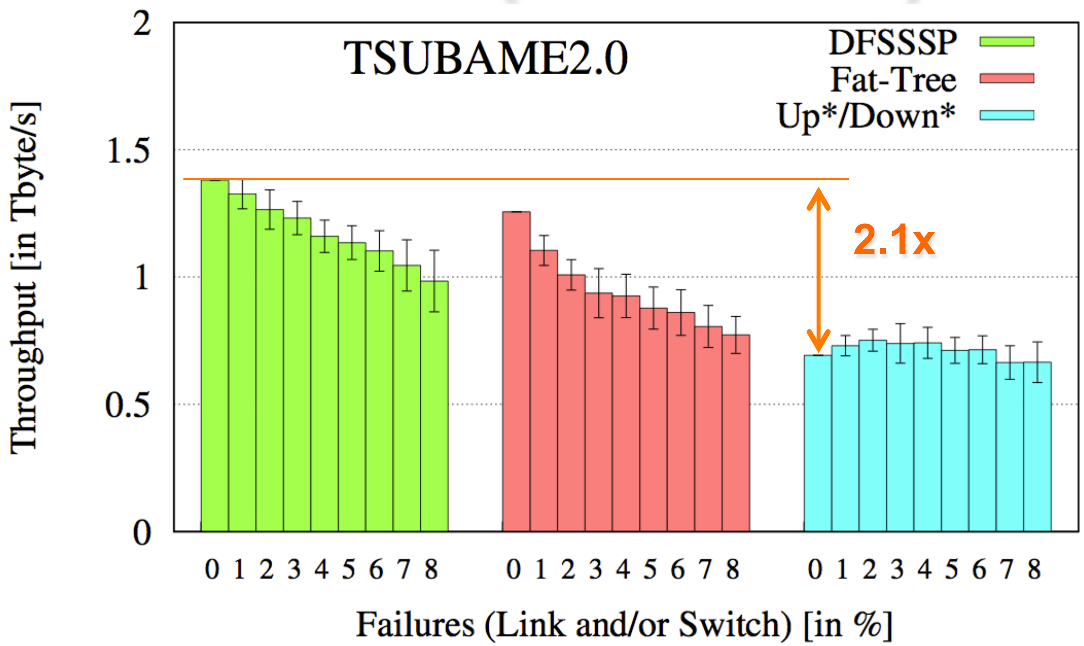


Different networks with ≈2,200 HCAs

TSUBAME2.0 (TiTech)

Up*/Down* routing is default on TSUBAME2.0

Changing to DFSSSP routing on TSUBAME2.0 improves the throughput by 2.1x for the fault- free network and increases TSUBAME's fail-in-place characteristics



Switch and link failures (1 : 13 ratio)

TABLE III. INTERCEPT, SLOPE, AND R^2 FOR TSUBAME2.0 (DEFAULT ROUTING: ITALIC; BEST ROUTING: BOLD)

Routing	Intercept [in Gbyte/s]	Slope	R^2
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
LASH	22.92	-0.01	0.10

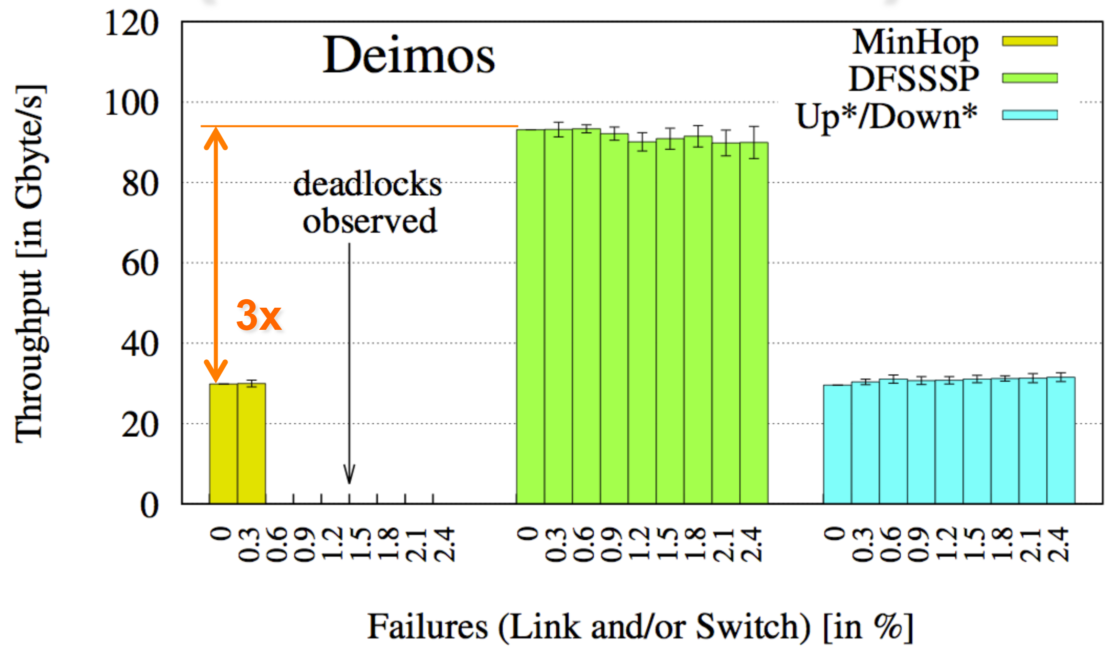
- **Simulation of 8 years of TSUBAME2.0's lifetime (≈1% annual link/switch failure)**
- **Upgrade TSUBAME2.0 to 2.5 did not change the network**
- **No correlation between throughput using Up*/Down* and failures**

Deimos (TU Dresden)

Improvement of 3x with DFSSSP over MinHop (default; deadlocks)

No degradation even with fail-in-place approach

➔ No maintenance cost (except for replacing critical components)

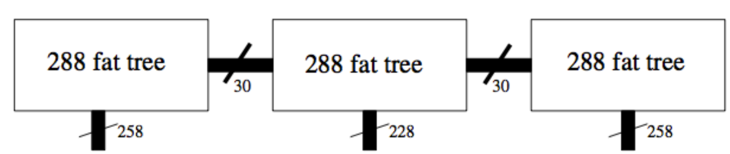


Switch and link failures (1 : 2 ratio)

- Sim. of 8 years of Deimos' lifetime (0.2% annual link & 1.5% switch failure)
- Deimos' network is very sparse

TABLE IV. INTERCEPT, SLOPE, AND R^2 FOR DEIMOS (DEFAULT ROUTING: ITALIC; BEST ROUTING: BOLD)

Routing	Intercept [in Gbyte/s]	Slope	R^2
MinHop	29.94	-	-
DFSSSP	93.40	-0.15	0.09
Up*/Down*	30.10	0.06	0.11
LASH	8.37	0.00	0.04



Deimos network topology

Presentation Overview

1. Topologies, Routing, Failures

2. Resilience Metrics

3. Simulation Framework

4. Influence of Failures

5. **Lessons Learned & Conclusions**

Toolchain Use Cases

Routing/Library Development

- Test new routings via plugin interface
- Improve MPI collectives to match oblivious routing

HPC Design

- Test topology/routing combinations
- Extrapolate throughput degradation over time based on estimated failure rates and derive operation policies

HPC System Management

- Simulate current throughput w/o influencing the real system and decide if maintenance/action is needed

Issues of curr. Routings

- **Topology-aware routing algorithms**
  - Few failures can have big influence on throughput
  - Resilience/deadlock issues for large #failures
  - Problems with unbalanced networks (e.g., thru adding management nodes, damaged HCAs, …)
- **Topology-agnostic routing algorithms**
  - Usually higher runtime ➔ recovery takes longer
  - Potentially lower throughput for some regular topologies
  - Scaling issues if deadlock-freedom is required (i.e., known DL-free routings, based on VLs, exceed available number of virtual lanes for large scale networks)

Concussion / Summery

What we can't give you

- **Name the best topology or the best routing algorithm**
- **Definitive answer which topology or routing is best for your needs**
- **General estimation on cost savings:**
  - Depends on many variables: such as network size, failure rate, hardware costs, maintenance costs, …

Concussion / Summery

However, we showed and can provide

- **Simulation framework helps to easily identify efficient topology/routing combination**
- **Toolchain** (see http://spcl.inf.ethz.ch/Research/Scalable_Networking/FIP)
  - Test system designs, topologies, routing algorithms
  - Evaluate throughput degradation of running system
- **Investigated routing algorithms (even fault-tolerant & topology-agnostic) show limitations**

→ **BUT: Fail-in-place networks are possible! ☺**

Acknowledgements

- **Eitan Zahavi (Mellanox)**
  – Developed the initial IBmodel for OMNeT++
- **Researchers at Simula Research Laboratory**
  – Ported the IB module to newest OMNeT++ version
- **HPC system administrators at Los Alamos National Lab, Technische Universität Dresden and Tokyo Institute of Technology**
  – Collected highly detailed failure data

References

[Banikazemi, 2008]: 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.

[Domke, 2011]: 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. Washington, DC, USA: IEEE Computer Society, May 2011, pp. 613–624.

[Flich, 2011]: 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.

References

[Gran, 2011]: 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.

[Ho, 1982]: 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.

[Hoefler, 2008]: 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.