Routing on the Channel Dependency Graph

SIAM PP'18, Waseda University, Tokyo, 2018-02-09

Satoshi MATSUOKA
Laboratory, GSIC, Tokyo Institute of Technology
Jens Domke, Dr.

Outline

- **Motivation**
- **Routing Deadlocks and Deadlock-Prevention Strategies**
  - Theorem of Dally and Seitz
  - Analytical Solution vs. Virtual Channels
  - Related Work: Comparison of existing Routing Algorithms
- **Routing on the Dependency Graph and Nue Routing for HPC**
  - Shortest-Path Routing **+** Virtual Channels **==** Deadlock-Freedom ?
  - Routing on the Dependency Graph
  - Nue Routing
- **Evaluation of Nue Routing**
  - Throughput Comparison for various Topologies
  - Runtime and Fault-tolerance of Nue
- **Summary and Conclusions**

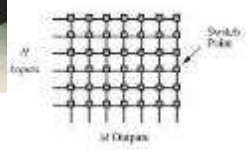
Motivation – Interconnection Networks for HPC-Systems

Towards ExaScale HPC

- ≥100.000 nodes [Kogge, 2008]
- Fat-trees not sustainable
- Sparse/random topologies (SimFly [Besta, 2014], Dragonfly [Kim, 2008], Jellyfish [Singla, 2012], etc.)

1993: NWT (NAL)
140 Nodes
Crossbar Network

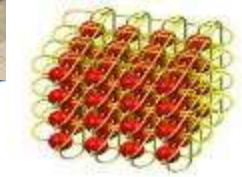
[F1]



[F2]

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

[F3]



[F4]

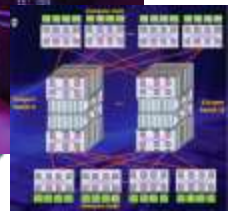
2011: K (RIKEN)
82,944 Nodes
6D Tofu Network

[F5]

[F6]

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

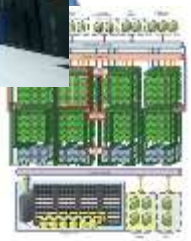
[F7]



[F8]

2016: Sunway TaihuLight
(NRCPC) 40,960 Nodes
Fat-Tree

[F9]



[F10]

Massive networks needed to connect all compute nodes of supercomputers (see TOP500 list)

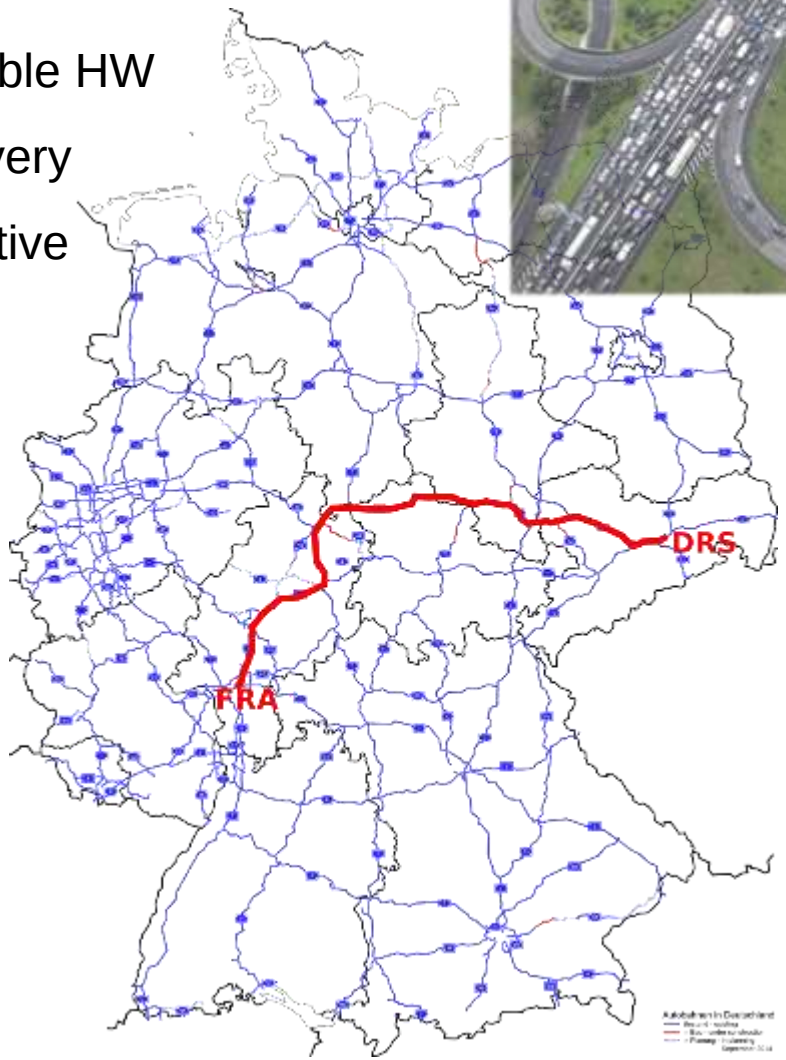
Motivation – Routing in HPC Network

- Similarities to car traffic, …
- Key metrics: low latency, high throughput, low congestion, fault-tolerant, deadlock-free, utilize (all) available HW
- Low runtimes for fast fault recovery
- Online/reactive vs. offline/proactive path calculation
- Flow-aware/dynamic vs. oblivious
- Static (or adaptive)

… and more

➥ Highly depended on network topology and technology

[F11]



[F12]

Motivation – Assumptions for the Remainder of the Talk

- Requirements and assumptions:
  - Network $\boldsymbol{I}$ consists of
    $I = G(N, C)$
    $with \;\; C \subseteq N \times N$
    - Switches S and terminals T, with $S \cup T = N$, connected by full-duplex channels/links C
  - Routing $\boldsymbol{R}$ should be
    $R(c_i, n_d) = c_{i+1}$
    $with \;\; n_d \in N \wedge c_i \in C$
    - Destination-based (and unicast)
    - Shortest-path and balanced
    - Deadlock-free (for lossless technologies)
    - Flow-oblivious and static
    - Support arbitrary topologies
  - Resources are limited
    - Compute power
    - Virtual channels (for deadlock-freedom)
  - Network topology can be
    - Regular or irregular
    - Faulty during operation

Outline

- **Motivation**
- **Routing Deadlocks and Deadlock-Prevention Strategies**
  - Theorem of Dally and Seitz
  - Analytical Solution vs. Virtual Channels
  - Related Work: Comparison of existing Routing Algorithms
- **Routing on the Dependency Graph and Nue Routing for HPC**
  - Shortest-Path Routing **+** Virtual Channels **==** Deadlock-Freedom ?
  - Routing on the Dependency Graph
  - Nue Routing
- **Evaluation of Nue Routing**
  - Throughput Comparison for various Topologies
  - Runtime and Fault-tolerance of Nue
- **Summary and Conclusions**

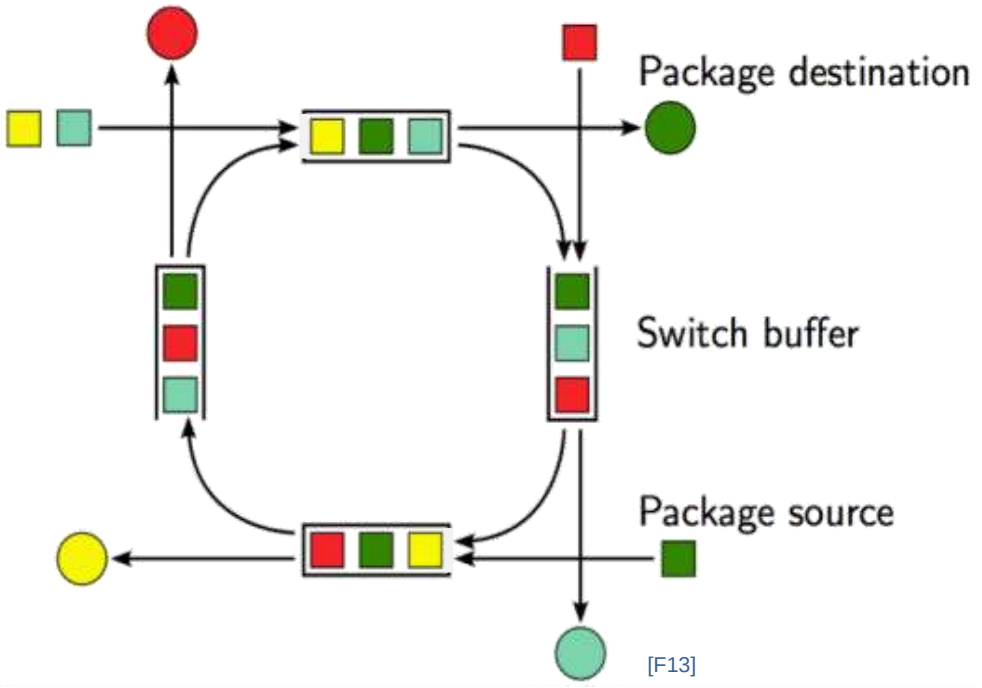
Routing Deadlocks – Credit Buffers in Lossless Interconn.

Deadlock [Coffman, 1971]

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.

Lossless interconnection network

- Switches use credit-based flow-control [Kung, 1994] and linear forwarding tables (LFTs)
- Messages forwarded only if receive-buffer available



(similar to deadlocks in wormhole-routed systems [Dally, 1987])

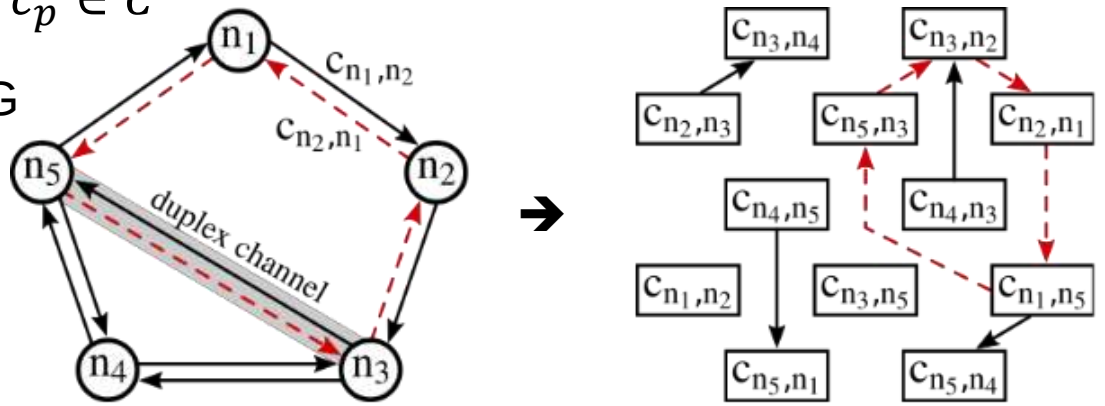
Routing Deadlocks – Channel Dependency Graph

Theorem of Dally and Seitz [Dally, 1987]

A routing algorithm for an interconnection network is deadlock-free, if and only if there are no cycles in the corresponding *channel dependency graph*.

Channel Dependency Graph (CDG)

- Channels/links of $I = G(N, C)$ are nodes in the CDG $D = G(C, E)$, with ordered pairs $(n_x, n_y) =: c_p \in C$
- Connect nodes of C of the CDG only if adjacent links are used to route messages, i.e.:
  $\exists\, n_y \in N: R(c_p, n_y) = c_q$



(a) Ring with shortcut (b) Corresponding (cyclic) CDG

Figure 11: Using a shortest-path, counter-clockwise routing for network I induces the channel dependency graph D

Routing Deadlocks – Ignoring, Preventing, Avoiding, …

- Ignoring routing deadlocks
  - "Resolving" via package lifetime [IBTA, 2015]
  - Fast path calculation (e.g., MinHop [MLX, 2013], SSSP [Hoefler, 2009])
- Deadlock-prevention via analytical solution
  - Topology-awareness required ➔ limited to subset of (non-faulty) topologies
  - Or avoid "bad" turns (e.g., Up*/Down* routing) ➔ poor path balancing [Flich, 2002]
- **Deadlock-prevention via virtual channels**
  - Allows good path balancing ➔ links/turns aren't limited [Domke, 2011]
  - Requires breaking cycles in the CDG ➔ higher time complexity
  - Virtual channels (VCs) are limited (e.g., max. of 15 in IB [Shanley, 2003])
- Others approaches, e.g.
  - Bubble Routing [Wang, 2013] ➔ not supported by current devices
  - Controller principle [Toueg, 1980] ➔ doesn't scale and currently not supported

Routing Deadlocks – Virtual Channels or Virtual Networks

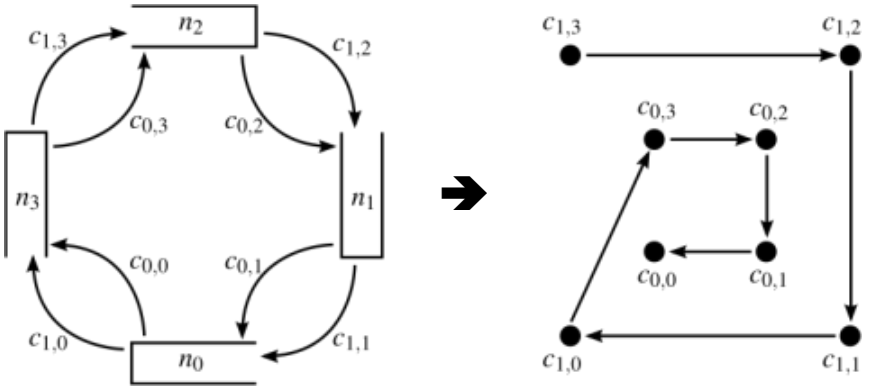
- Virtual channels == multiple sets of individually managed credit buffers in one port [Dally, 2003]

➥ Split channels/links into multiple virtual channels

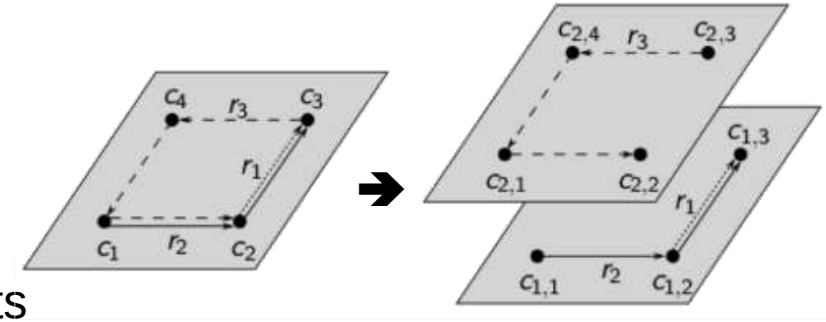
➥ Use different channels to generate acyclic channel dependency graph D

- **Version 1** (virtual channel transitioning)
  - packets can switch between ‘high’ and ‘low’ channel [Dally, 1987]
- **Version 2** (combine into virtual layers)
  - ‘high’ channels build ‘high’ layer and packets stay within one layer [Skeie, 2002]
- VCs are limited due to implementation costs (control logic, physical buffer size, etc.)

Version 1



Version 2



[F14]

Related Work: Comparison of existing Routing Algorithms

Routing	Network $I = G(N, C)$	Latency	Through-put	DL - Freedom	#VC	FT	Time Complexity[#]
DOR [Rauber, 2010]	meshes	+	+	yes	1	no	N/A
Torus-2QoS [MLX, 2013]	2D/3D meshes/tori	+	+ +	yes	≥ 2	limited	N/A
Fat-Tree [Zahavi, 2010]	k-ary n-tree	+	+ +	yes	1	limited	N/A
MinHop [MLX, 2013]	arbitrary	+	+	no	1	yes	$\mathcal{O}(\lvert N\rvert \cdot \lvert C\rvert)$
Up*/Down* [Schroeder, 1991]	arbitrary	- -	- -	yes	1	yes	$\mathcal{O}(\lvert N\rvert \cdot \lvert C\rvert)$
MUD [Flich, 2002]	arbitrary**	-	-	yes	≥ 2	yes	$\mathcal{O}(\lvert N\rvert \cdot \lvert C\rvert)$
(DF)SSSP [Domke, 2011; Hoefler, 2009]	arbitrary	+	+ +	(yes*) no	(≥)1	yes	$\mathcal{O}(\lvert N\rvert^2 \cdot log\,\lvert N\rvert)$
L-turn [Koibuchi, 2001]	arbitrary	-	-	yes	1	yes	$\mathcal{O}(\lvert N\rvert^3)$
LASH [Skeie, 2002]	arbitrary	+	-	yes*	≥ 1	yes	$\mathcal{O}(\lvert N\rvert^3)$
LASH-TOR [Skeie, 2004]	arbitrary**	-	-	yes	≥ 1	yes	$\mathcal{O}(\lvert N\rvert^3)$
SR [Mejia, 2006]	arbitrary	-	-	yes	1	yes	$\mathcal{O}(\lvert N\rvert^3)$
Smart [Cherkasova, 1996]	arbitrary	-	+	yes	1	yes	$\mathcal{O}(\lvert N\rvert^9)$
BSOR(M) [Kinsy, 2009]	arbitrary**	+	++[†]	yes	≥ 1	yes	N/A

[#]: to (re-)calculate all LFTs for network I [Flich, 2012]
*: limited; might exceed available #VCs
**: not easily applicable for destination-based forwarding
[†]: requ. knowledge of bandwidth demands

Outline

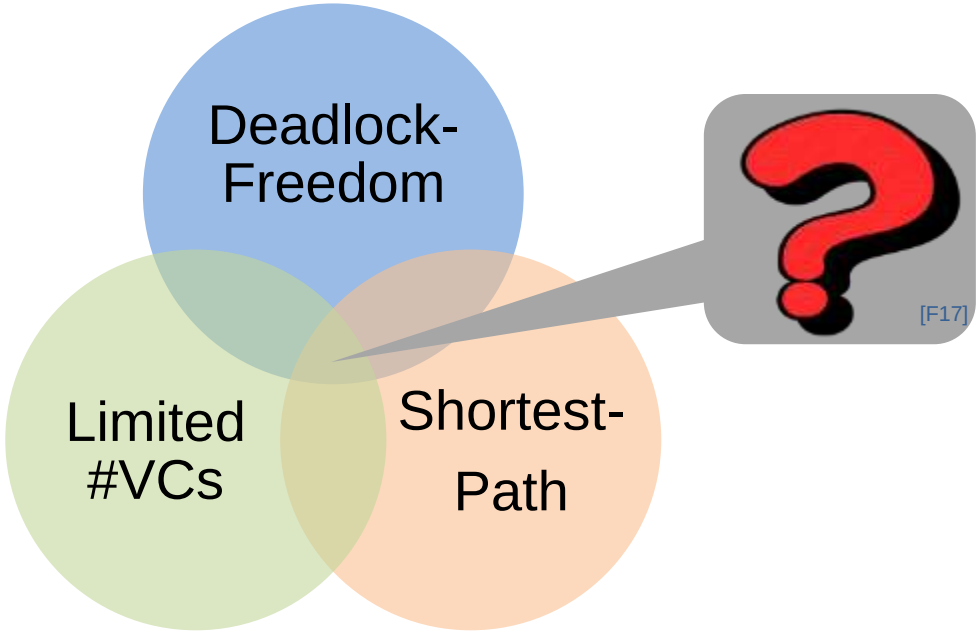
- **Motivation**
- **Routing Deadlocks and Deadlock-Prevention Strategies**
  - Theorem of Dally and Seitz
  - Analytical Solution vs. Virtual Channels
  - Related Work: Comparison of existing Routing Algorithms
- **Routing on the Dependency Graph and Nue Routing for HPC**
  - Shortest-Path Routing **+** Virtual Channels **==** Deadlock-Freedom ?
  - Routing on the Dependency Graph
  - Nue Routing
- **Evaluation of Nue Routing**
  - Throughput Comparison for various Topologies
  - Runtime and Fault-tolerance of Nue
- **Summary and Conclusions**

Routing Deadlocks – Deadlock-Freedom & Shortest-Path

Can one ensure deadlock-freedom, while enforcing shortest-path routing?

Assuming the following:

- Arbitrary topology and arbitrary but fixed number of VCs (0/1, 2, or more…)
- Routed by destination-based routing algorithm

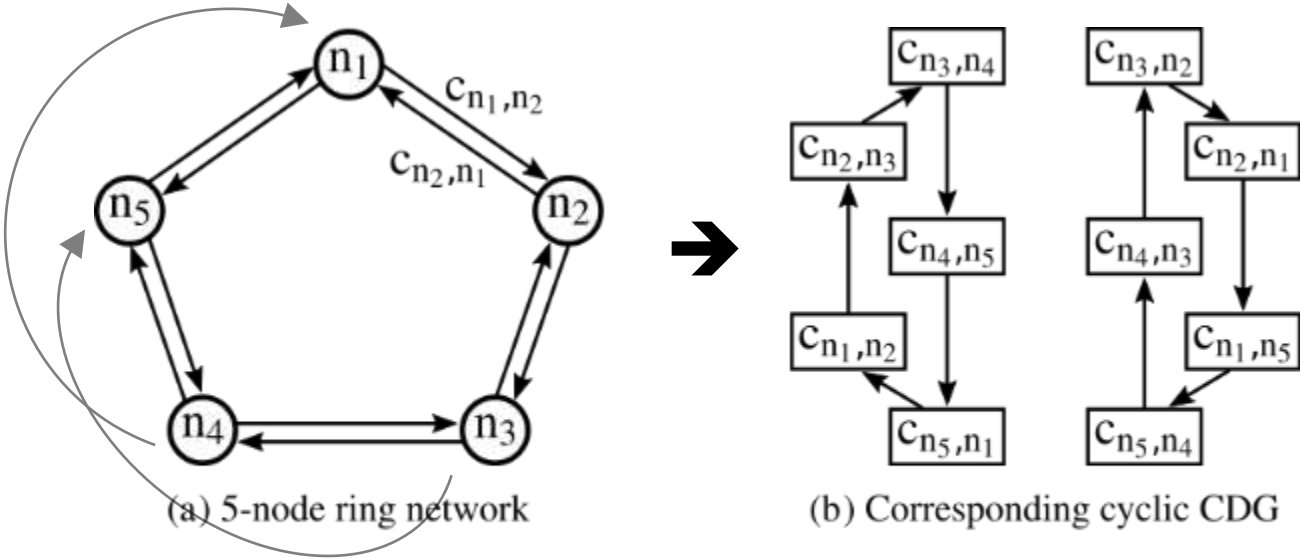


Routing Deadlocks – Deadlock-Freedom & Shortest-Path

Easy **counter example**, assume:

- Ring network with 5 nodes; no/one virtual channels; shortest-path routing
- Node n_1 sends messages to n_3; n_2 sends to n_4; n_3 sends to n_5; …

➥ CDG is cyclic ➔ routing is NOT deadlock-free (Theorem of Dally and Seitz)



(a) 5-node ring network (b) Corresponding cyclic CDG

Proposition

Assuming a limited number of virtual channels, then it can be impossible to remove all cycles from a channel dependency graph, which is induced by a shortest-path routing algorithm.

Outline

- **Motivation**
- **Routing Deadlocks and Deadlock-Prevention Strategies**
  - Theorem of Dally and Seitz
  - Analytical Solution vs. Virtual Channels
  - Related Work: Comparison of existing Routing Algorithms
- **Routing on the Dependency Graph and Nue Routing for HPC**
  - Shortest-Path Routing **+** Virtual Channels **==** Deadlock-Freedom ?
  - **Routing on the Dependency Graph**
  - Nue Routing
- **Evaluation of Nue Routing**
  - Throughput Comparison for various Topologies
  - Runtime and Fault-tolerance of Nue
- **Summary and Conclusions**

Routing on the Channel Dependency Graph

Analytical Solution / Turn Model

Step1: restriction of possible turns in I

Step2: calculate (non-shortest) paths in I

➥ overly restrictive; poor balancing

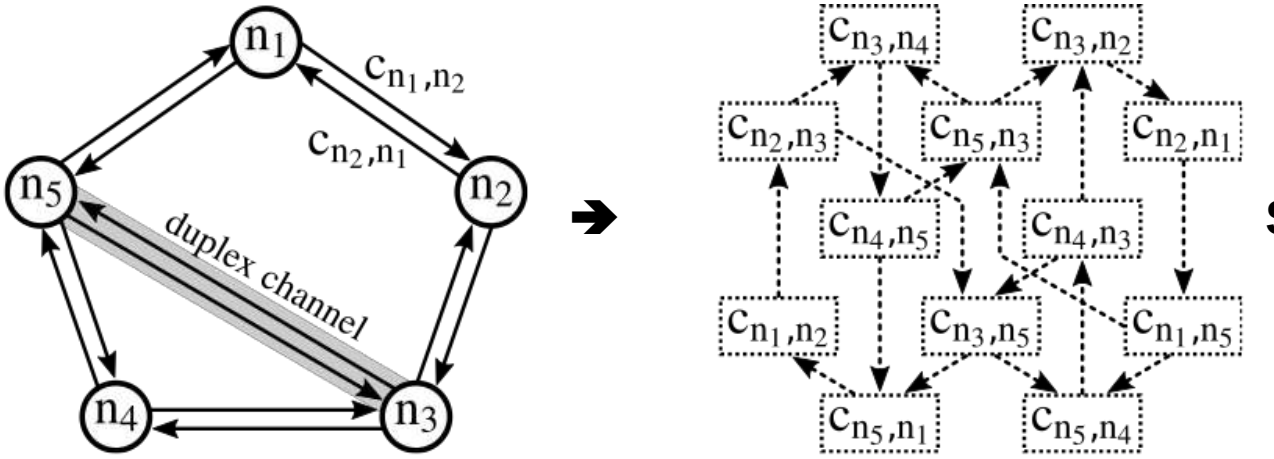
Virtual Channel Approach

Step1: calculate shortest paths in I

Step2: create acyclic CDG per virtual layer

➥ needed #VCs is unbound

Combine graph representation of network I and CDG D into a supergraph and calculate routing in ”one step”



(a) 5-node ring network with shortcut

(b) Corresponding complete CDG $\overline{D}$

Supergraph
Complete Channel Dependency Graph $\overline{D}$

Figure 13: Complete CDG $\overline{D}$ for the 5-ring network with shortcut

The Complete Channel Dependency Graph

From the routing-dependent CDG D to the independent complete CDG $\overline{D}$

- Complete CDG $\overline{D} := G(C, \bar{E})$, with $\bar{E} \subseteq C \times C$ is defined by

$$\forall (n_x, n_y), (n_y, n_z) \in C, n_x \neq n_z : \big((n_x, n_y), (n_y, n_z)\big) \in \bar{E}$$

Definition

$\overline{D}$ is cycle-free $\Leftrightarrow$ $D \subseteq \overline{D}$ is acyclic for any D induced by a given routing R

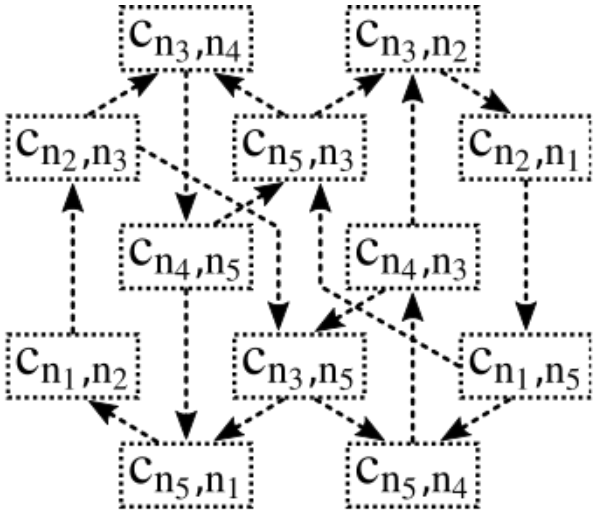


Figure 14: Complete CDG $\overline{D}$ for the 5-node ring network with shortcut

Advantages of the complete CDG over network graph

- Includes node/link information
- Includes all possible routes (i.e., all available channel dependencies)
- Allows “on-demand” checks for acyclic subgraphs

Routes in the Complete Channel Dependency Graph

- *Initially:* all edges in $\bar{E}$ are in ***unused*** state (hence, no routing R applied, yet)
- *Step1:* Route from n_3 to n_4 via node n_5 ➔ changes edge $(c_{n_3,n_5}, c_{n_5,n_4})$ into ***used*** state
- *Step2:* Route from n_5 to n_3 via n_4
- *Step3:* Route from n_4 to n_5 via n_3? ➔ closes cycle in $\bar{D}$
  ➥ mark edge ***blocked*** ➔ use alternative, direct route c_{n_4,n_5}

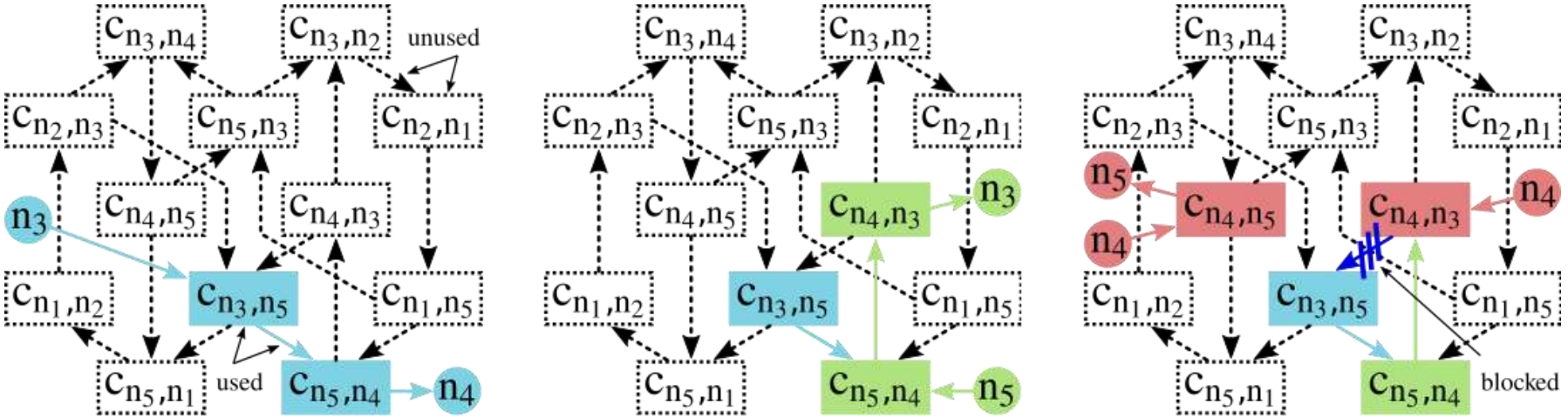


Figure 15: Sketching the process of finding paths between network nodes within the complete CDG $\bar{D}$ while avoiding to close any cycles in the induced CDG D

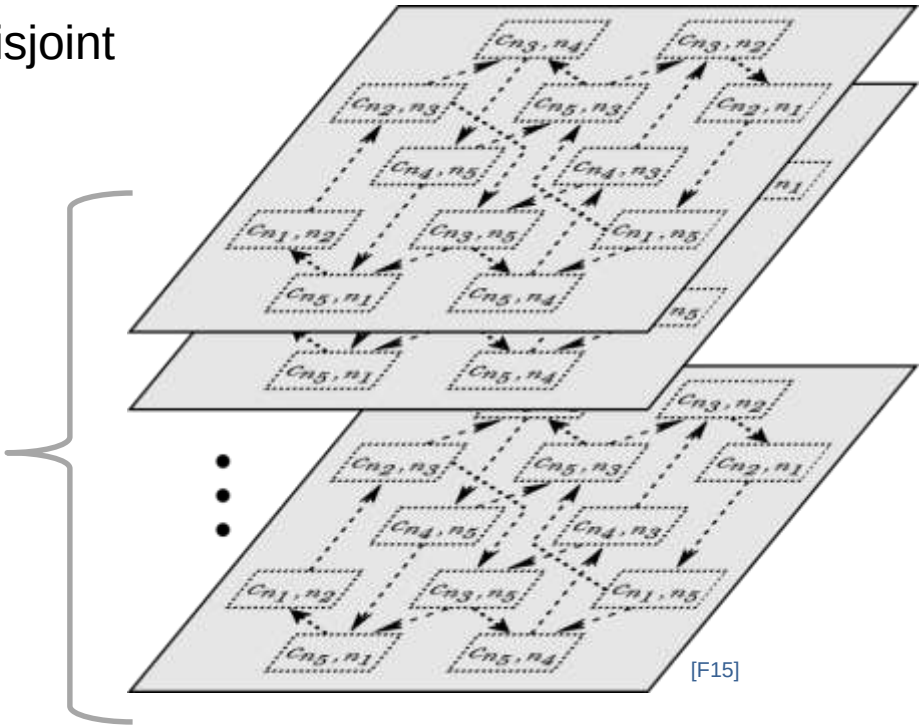
Outline

- **Motivation**
- **Routing Deadlocks and Deadlock-Prevention Strategies**
  - Theorem of Dally and Seitz
  - Analytical Solution vs. Virtual Channels
  - Related Work: Comparison of existing Routing Algorithms
- **Routing on the Dependency Graph and Nue Routing for HPC**
  - Shortest-Path Routing **+** Virtual Channels **==** Deadlock-Freedom ?
  - Routing on the Dependency Graph
  - **Nue Routing**
- **Evaluation of Nue Routing**
  - Throughput Comparison for various Topologies
  - Runtime and Fault-tolerance of Nue
- **Summary and Conclusions**

Create Multiple Virtual Networks and Assign Destinations

Nue's goal: find deadlock-free routes between each pair of nodes in I while complying with the VC limitation

- Partition node set N into $k := \#VCs$ disjoint subsets (e.g., w/ METIS [Karypis, 1998])
  - ➥ destinations N_i^d, with $1 \leq i \leq k$, for routes towards $n \in N_i^d$
- Create k complete CDGs $\overline{D}_i$ (virtual supergraphs) and assign one destination set to each
- Calculate routes from all (source) nodes to all destinations N_i^d within each complete CDG $\overline{D}_i$ (w/o closing a cycle)



➥ Each CDG is acyclic ➔ Nue routing is deadlock-free

Dijkstra's Algorithm and Weight Updates for Balancing

Destination-based Routes

- Use modified Dijkstra's algorithm on complete CDG $\overline{D}$ (similar to (DF-)SSSP routing on I)
- Destination $n_0 \in N_i^d$ acts as source node for Algorithm 2
- **Main difference:** use edge if and only if no cycle is created

Algorithm 2: Dijkstra's Algorithm within $\overline{D}$

Input: $I = G(N, C)$, $\overline{D} = G(C, \overline{E})$, source $n_0 \in N$
Result: P_{n_y, n_0} for all $n_y \in N$ (and $\overline{D}$ is cycle-free)

```
1  foreach node n ∈ N do
2      n.distance ← ∞
3      n.usedChannel ← ∅
4  n_0.distance ← 0
5  c_0.distance ← 0
6  FibonacciHeap Q ← {c_0}
7  while Q ≠ ∅ do
8      c_p ← Q.findMin()
9      foreach (c_p, c_q) ∈ Ē with (c_p, c_q).state ≠ blocked do
           // Let n_{c_q} ∈ N be the tail of directed channel c_q
10         if c_p.distance + c_q.weight < n_{c_q}.distance then
11             (c_p, c_q).state ← used                    // modifies D̄
12             if D̄ is cycle-free then
13                 Q.add(c_q)
14                 c_q.distance ← c_p.distance + c_q.weight
15                 n_{c_q}.distance ← c_p.distance + c_q.weight
16                 n_{c_q}.usedChannel ← c_q
17             else
18                 (c_p, c_q).state ← blocked
```

Path balancing

- Use weights for channels (additionally to node distances)
- Update channel weights of used links after Algorithm 2 finished

➥ Minimizes overlapping of routes if possible

Checking for Absence of Cycles in the Complete CDG

Nue really checks every edge?

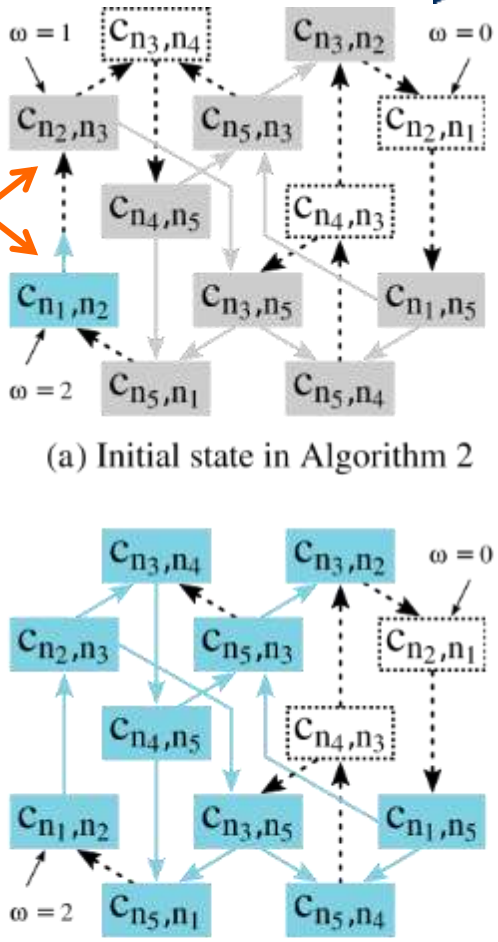
- New **subgraph identification** number ω for each call to Dijkstra's on $\overline{D}$ (Algorithm 2)
- ω gets assigned to each node/edge of $\overline{D}$ identifying connected and **acyclic** subgraphs D

$$\omega : C \cup \overline{E} \to \mathbb{Z}_0^+ \cup \{-1\}, \text{ with}$$

$$\omega(x) = \begin{cases} -1 & \text{if } D \cup x \text{ form cycle in } \overline{D}, \text{ i.e., } x \text{ is } \textit{blocked}, \\ 0 & \text{if } x \notin D, \text{ i.e., } x \text{ is } \textit{unused}, \\ \geq 1 & \text{if } x \text{ is in the } \textit{used} \text{ state} \end{cases}$$

➥ No check for $e \in \overline{E}$
- $\omega(e) = -1$, i.e. ***blocked***
- $\omega(e) \geq 1$, already ***used***
- $\omega(e) = 0$, but **merging two** different acyclic D ➥ acyclic

➥ Cycle check needed
- $\omega(e) = 0$ and same ω for adjacent nodes $c \in C$



(a) Initial state in Algorithm 2

(b) State after five iterations

Figure 16: State change of $\overline{D}$ after five iterations with Algorithm 2, starting from c_{n_1,n_2}

Routing Impasse and Fallback to Escape Paths

Impasse problem

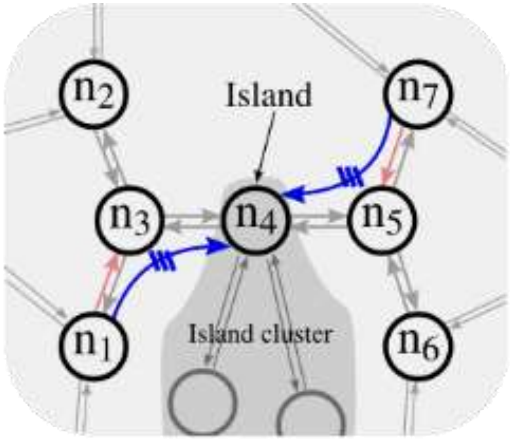
- Iterative path calculation within $\overline{D}$ can get stuck
  - ➥ Not all nodes discoverable [Cherkasova, 1996]

Possible solutions

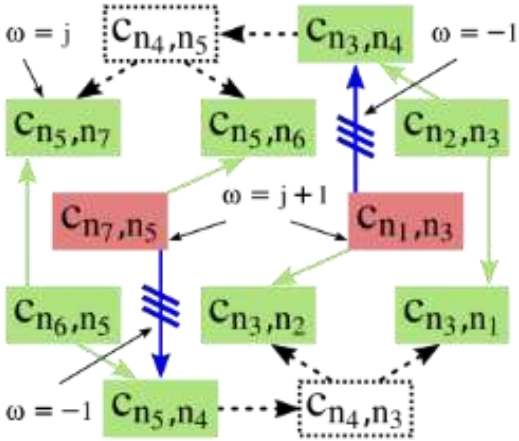
- Backtracking (similar to 8-queens problem, w/ $\#q \gg 8$)
  - ➥ Very expensive in term of runtime
- Fallback to **escape paths** (initial set of ***used*** channel dependencies which cannot be mark as blocked)
  - ➥ Many impasses for large topologies

Nue's approach

- Use local backtracking (max. 2 hops away) and only fallback to escape paths if necessary
  - ➥ Local backtracking works for most impasses
  - ➥ Very time- and memory efficient



(a) Subnetwork I* of a larger network I



(b) Intermediate state of the complete CDG $\overline{D}$

Figure 17: Impasse of Algorithm 2 to reach n_4 based on previously placed routing restrictions

Pseudo Code of Nue Routing

Algorithm 3: Nue routing calculates all paths within a network I for a given number of virtual channels $k \geq 1$

Input: $I = G(N, C), k \in \mathbb{N}$
Result: Path P_{n_x, n_y} for all $n_x, n_y \in N$

Partition N into k disjoint subsets $N_1^d, \ldots, N_k^d$ of destinations
foreach *Virtual layer* L_i *with* $i \in \{1, \ldots, k\}$ **do**
    `// Check attached comments for details about each step`
    Select a subset of nodes $N_i^d \subseteq N$ for virtual layer L_i
    Create a convex subgraph H_i for N_i^d
    Identify central $n_{r,i} \in N_i^H$ of H_i
    Create a new complete CDG $\overline{D}_i$
    Define escape paths D_i^s for root $n_{r,i}$
    foreach *Node* $n \in N_i^d$ **do**
        Identify deadlock-free paths $P_{\cdot,n}$
        Store these paths, e.g., in forwarding tables
        Update channel weights in $\overline{D}_i$ for these paths

Outline

- **Motivation**
- **Routing Deadlocks and Deadlock-Prevention Strategies**
  - Theorem of Dally and Seitz
  - Analytical Solution vs. Virtual Channels
  - Related Work: Comparison of existing Routing Algorithms
- **Routing on the Dependency Graph and Nue Routing for HPC**
  - Shortest-Path Routing **+** Virtual Channels **==** Deadlock-Freedom ?
  - Routing on the Dependency Graph
  - Nue Routing
- **Evaluation of Nue Routing**
  - Throughput Comparison for various Topologies
  - Runtime and Fault-tolerance of Nue
- **Summary and Conclusions**

Simulation Framework and Simulated Topologies

- Flit-level simulation framework for IB (OMNet++ [Varga, 2008] & ibmodel [Gran, 2011])
- Communication throughput of all-to-all traffic pattern (similar to MPI_Alltoall) with 2 KiB messages
- Multiple topologies with approx. 1,000 compute nodes (or terminals)
- Comparison of Nue to all routing algorithms implemented in OFED OpenSM (if applicable to the topology)
- Networks configured as 4xQDR IB with 36-port switches (48-p for Cascade) and 8 virtual channels
- Nue simulations for 1VC, …, 8VCs

Table 1: Topology configurations (w/ link redundancy r) used for throughput simulations in Fig. 10

Topology	Switches	Terminals	Channels	r
Random	125	1,000	1,000	1
6x5x5 3D-Torus	150	1,050	1,800	4
10-ary 3-tree	300	1,100	2,000	1
Kautz $(d = 7, k = 3)$	150	1,050	1,500	2
Dragonfly $(a = 12, p = 6, h = 6, g = 15)$	180	1,080	1,515	1
Cascade (2 groups)	192	1,536	3,072	1
Tsubame2.5	243	1,407	3,384	1

Throughput Comparison for various Topologies

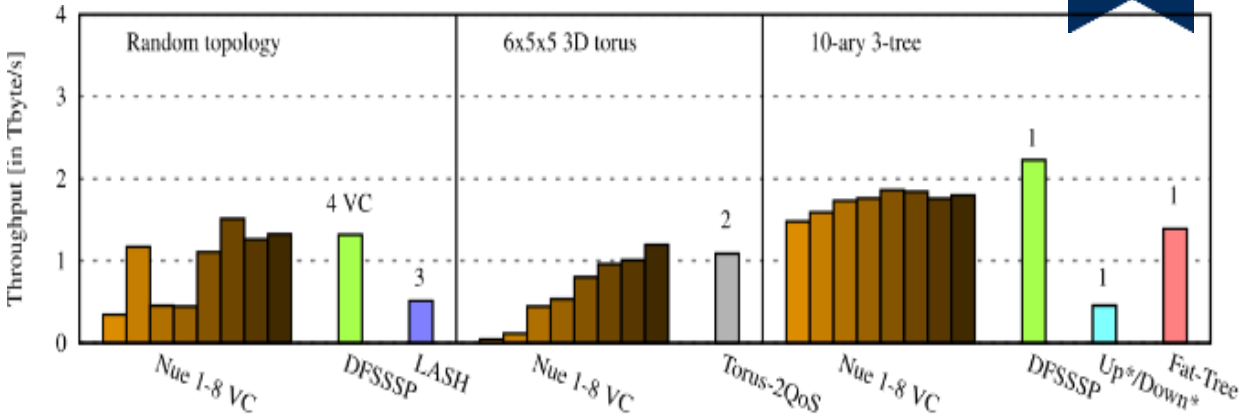
- Throughput shown (higher is better)
- #VCs used by routing listed above bars

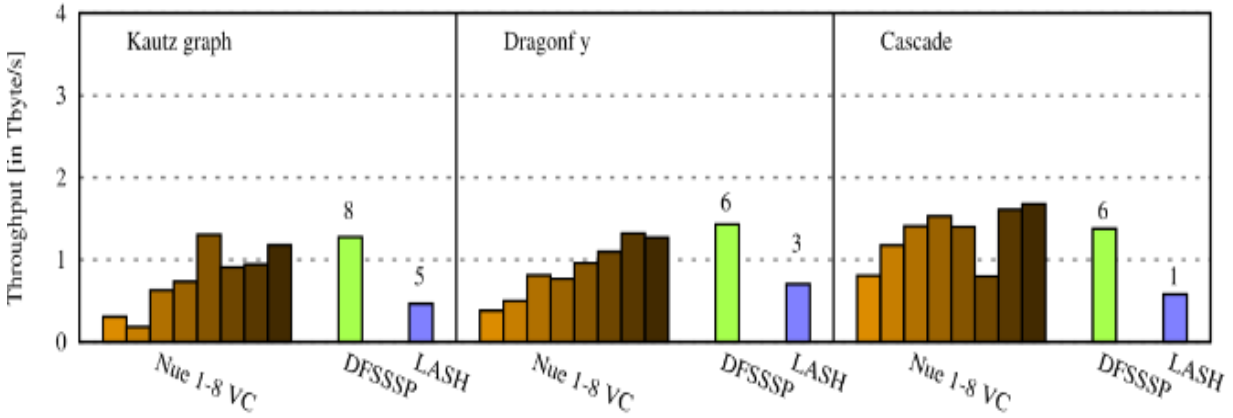
Results

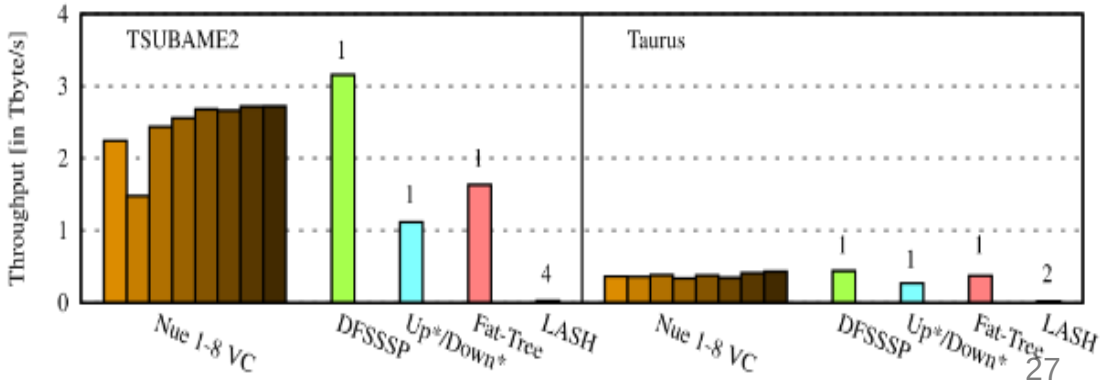
➥ Nue offers competitive performance (between 83.5% (10-ary 3-tree) and 121.4% (Cascade))

➥ Achievable throughput for Nue grows with available/used #VCs

➥ Only downside: high number of fallbacks to escape paths can cause worse path balancing
➥ diminished throughput







Runtime and Fault-tolerance of Nue Routing

Nue implemented in OpenSM

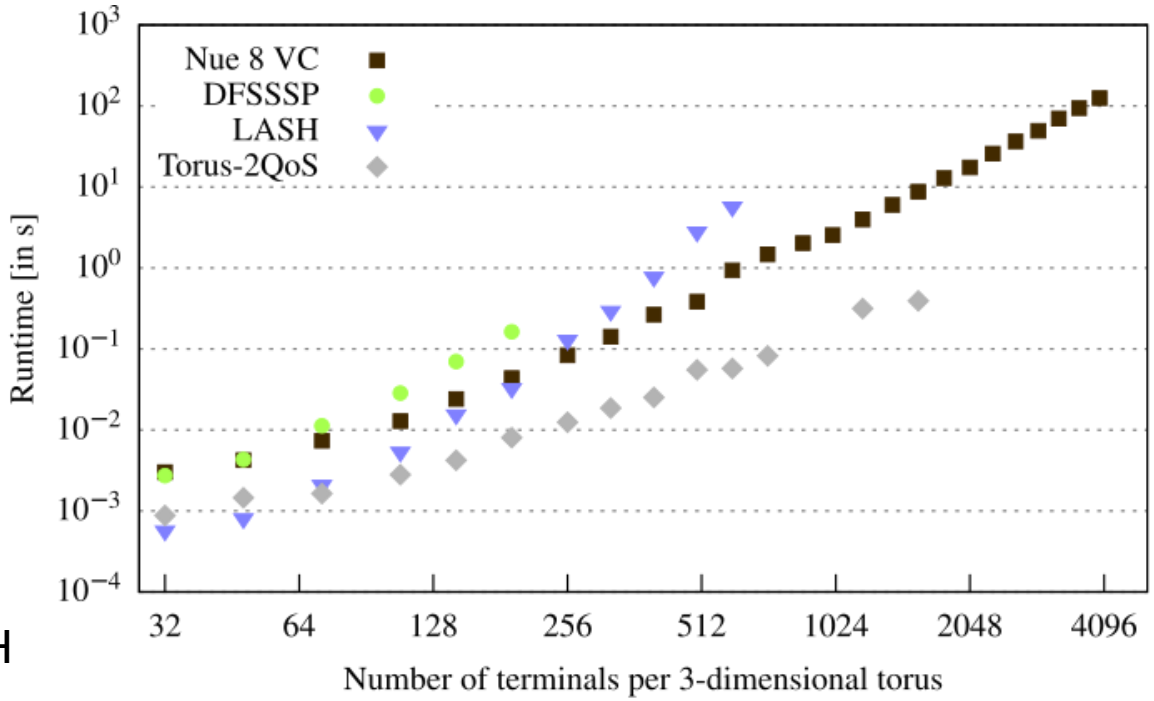
- Patched OpenSM ntegrated in fail-in-place toolchain for fair runtime comparison
- Created 25 3D torus networks (size: 2x2x2, 2x2x3, 2x3x3,…, 10x10x10) with 4 terminal nodes per switch; InfiniBand with 8 VCs
- Inject 1% random link/channel failures (cf. annual failure rate)

Expected results

- DFSSSP/LASH run out of VCs (➔ not deadlock-free)
- Torus-2QoS limited fault-tolerance
- Nue is always applicable

Unexpected results

- Faster routing calculation with Nue vs. DFSSSP/LASH



Outline

- **Motivation**
- **Routing Deadlocks and Deadlock-Prevention Strategies**
  - Theorem of Dally and Seitz
  - Analytical Solution vs. Virtual Channels
  - Related Work: Comparison of existing Routing Algorithms
- **Routing on the Dependency Graph and Nue Routing for HPC**
  - Shortest-Path Routing **+** Virtual Channels **==** Deadlock-Freedom ?
  - Routing on the Dependency Graph
  - Nue Routing
- **Evaluation of Nue Routing**
  - Throughput Comparison for various Topologies
  - Runtime and Fault-tolerance of Nue
- **Summary and Conclusions**

Summary – Features of destination-based Nue Routing

Routing	Network $I = G(N, C)$	Latency	Through-put	DL-Freedom	#VC	FT	Time Complexity[#]
DOR [Rauber, 2010]	Meshes	+	+	yes	1	no	N/A
Torus-2QoS [MLX, 2013]	2D/3D meshes/tori	+	+ +	yes	≥ 2	limited	N/A
Fat-Tree [Zahavi, 2010]	k-ary n-tree	+	+ +	yes	1	limited	N/A
MinHop [MLX, 2013]	arbitrary	+	+	no	1	yes	$\mathcal{O}(\lvert N\rvert \cdot \lvert C\rvert)$
• • •							
LASH [Skeie, 2002]	arbitrary	+	-	yes*	≥ 1	yes	$\mathcal{O}(\lvert N\rvert^3)$
LASH-TOR [Skeie, 2004]	arbitrary**	-	-	yes	≥ 1	yes	$\mathcal{O}(\lvert N\rvert^3)$
SR [Mejia, 2006]	arbitrary	-	-	yes	1	yes	$\mathcal{O}(\lvert N\rvert^3)$
Smart [Cherkasova, 1996]	arbitrary	-	+	yes	1	yes	$\mathcal{O}(\lvert N\rvert^9)$
BSOR(M) [Kinsy, 2009]	arbitrary**	+	++†	yes	≥ 1	yes	N/A
Nue Routing	**arbitrary**	**+**	**+/++**	**yes**	**≥ 1**	**yes**	$\mathcal{O}(\lvert N\rvert^2 \cdot log\ \lvert N\rvert)$

[#]: to (re-)calculate all LFTs for network I [Flich, 2012]
**: not easily applicable for destination-based forwarding
*: limited; might exceed available #VCs
†: requ. knowledge bandwidth demands

Conclusions

- Routing on the complete CDG: Nue demonstrates new approach to avoid deadlocks with limited VC resources (➔ template for new strategies)
- First algorithm to guarantee DL-freedom for arbitrary but fixed #VCs (incl. w/o available VCs)
  - ➥ Combining Quality-of-Service (QoS) and deadlock-freedom for IB
- Offers competitive bandwidth/latency and path calculation time
  - Throughput from 83.5% to 121.4% for all-to-all traffic pattern compared to best routing
  - Low time complexity $\mathcal{O}(|N|^2 \cdot log\,|N|)$ and memory complexity $\mathcal{O}(|N|^2)$
- Applicable to statically routed technologies (e.g., IB, OPA, RoCE, …)
- Since Jan. 2018 in official & open-source IB stack of the OpenFabrics Alliance

http://git.openfabrics.org/?p=~halr/opensm.git

(or simply wait for next OpenSM release version 3.3.21)

Tokyo Tech

[F16]

Nue – Japanese chimera combining the advantages of existing routing algorithms

http://domke.gitlab.io/ domke.j.aa@m.titech.ac.jp

References

[Alverson, 2012] B. Alverson, E. Froese, L. Kaplan, and D. Roweth. Whitepaper: Cray XC™ Series Network. Tech. rep. WP-Aries01-1112. Cray Inc., Nov. 2012, p. 28. URL: http://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf

[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“. In: SIGOPS Oper. Syst. Rev. 42.1 (Jan. 2008), pp. 54–62.

[Besta, 2014] M. Besta and T. Hoefler, "Slim Fly: A Cost Effective Low-Diameter Network Topology," New Orleans, LA, USA, 2014.

[Birrittella, 2015] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Under- wood, and R. C. Zak. „Intel Omni-path Architecture: Enabling Scalable, High Performance Fabrics“. In: 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects (HOTI). Santa Clara, CA: IEEE, Aug. 2015, pp. 1–9.

[Blake, 2007] D. Blake and W. Gore. Effect of Passive and Active Copper Cable Interconnects on Latency of Infiniband DDR compliant systems. Sept. 2007.

[Cherkasova, 1996] L. Cherkasova, V. Kotov, and T. Rokicki. „Fibre channel fabrics: evaluation and design“. In: Proceedings of the 29th Hawaii International Conference on System Sciences. Vol. 1. Jan. 1996, pp. 53–62.

[Coffman, 1971] E. G. Coffman Jr., M. Elphick, and A. Shoshani. „System Deadlocks“. In: ACM Computing Surveys 3.2 (1971), pp. 67–78.

[Boden, 2015] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. „Myrinet: a gigabit-per-second local area network“. In: IEEE Micro 15.1 (Feb. 1995), pp. 29–36.

[Dally, 1987] W. J. Dally and C. L. Seitz. „Deadlock-Free Message Routing in Multiprocessor Interconnection Networks“. In: IEEE Trans. Comput. 36.5 (1987), pp. 547–553.

[Dally, 2003] W. Dally and B. Towles. Principles and Practices of Interconnection Networks. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[Derradji, 2015] S. Derradji, T. Palfer-Sollier, J.-P. Panziera, A. Poudes, and F. W. Atos. „The BXI Interconnect Architecture“. In: Proceedings of the 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects. HOTI ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 18–25.

[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 (IPDPS). Washington, DC, USA: IEEE Computer Society, May 2011, pp. 613–624. ISBN: 0-7695-4385-7.

[Flich , 2002] J. Flich, P. López, J. C. Sancho, A. Robles, and J. Duato. „Improving InfiniBand Routing Through Multiple Virtual Networks“. In: Proceedings of the 4th International Symposium on High Performance Computing. ISHPC ’02. London, UK, UK: Springer-Verlag, 2002, pp. 49–63.

[Flich, 2012] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. López, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho. „A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms“. In: IEEE Transactions on Parallel and Distributed Systems 23.3 (Mar. 2012), pp. 405–425. ISSN: 1045-9219.

[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. SIMUTools ’11. ICST, Brussels, 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“. In: IEEE Transactions on Software Engineering SE-8.6 (1982), pp. 554–557.

[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.

References

[Hoefler, 2009] 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.

[Hoefler, 2011] T. Hoefler and M. Snir. „Generic Topology Mapping Strategies for Large-scale Parallel Architectures“. In: Proceedings of the 2011 ACM International Conference on Supercomputing (ICS'11). Tucson, AZ: ACM, June 2011, pp. 75–85.

[IBTA, 2015] InfiniBand Trade Association. InfiniBand™ Architecture Specification Volume 1 Release 1.3 (General Specifications). Mar. 2015.

[Karypis, 1998] G. Karypis and V. Kumar. „Multilevel K-way Partitioning Scheme for Irregular Graphs“. In: J. Parallel Distrib. Comput. 48.1 (Jan. 1998), pp. 96–129.

[Kim, 2008] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," SIGARCH Comput. Archit. News, vol. 36, no. 3, pp. 77-88, Jun. 2008.

[Kinsy, 2009] M. A. Kinsy, M. H. Cho, T. Wen, E. Suh, M. van Dijk, and S. Devadas. „Application-aware deadlock-free oblivious routing“. In: Proceedings of the 36th annual International Symposium on Computer Architecture. ISCA '09. New York, NY, USA: ACM, 2009, pp. 208– 219.

[Kogge, 2008] P. Kogge, K. Bergman, and S. Borkar, "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems," University of Notre Dame, Department of Computer Science and Engineering, Notre Dame, Indiana, TR-2008-13, Sep. 2008.

[Koibuchi, 2001] M. Koibuchi, A. Funahashi, A. Jouraku, and H. Amano. „L-turn routing: an adaptive routing in irregular networks“. In: International Conference on Parallel Processing. Sept. 2001, pp. 383–392.

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

[Mejia, 2006] A. Mejia, J. Flich, J. Duato, S.-A. Reinemo, and T. Skeie. „Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori“. In: 20th International Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 2006, p. 10.

[MLX, 2013] Mellanox Technologies. Mellanox OFED for Linux User Manual Rev. 2.0-3.0.0. Aug. 2013. URL: http://www.mellanox.com/related-docs/prod_software/Mellanox_OFED_ Linux_User_Manual_v2.0-3.0.0.pdf

[Rauber, 2010] T. Rauber and G. Rünger. Parallel Programming - for Multicore and Cluster Systems. Springer, 2010.

[Schroeder, 1991] M. D. Schroeder, A. Birell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker. „Autonet: A High-speed, Self-Configuring Local Area Network Using Point-to-Point Links“. In: IEEE Journal on Selected Areas in Communications 9.8 (Oct. 1991).

[Shanley, 2003] T. Shanley, J. Winkles, and I. MindShare. InfiniBand Network Architecture. PC System Architecture Series. Pearson Addison Wesley Prof, 2003.

[Singla, 2012] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking Data Centers Randomly," in Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), San Jose, CA, 2012, pp. 225-238.

[Skeie, 2002] T. Skeie, O. Lysne, and I. Theiss. „Layered Shortest Path (LASH) Routing in Irregular System Area Networks“. In: IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium. Washington, DC, USA: IEEE Computer Society, 2002, p. 194.

[Skeie, 2004] T. Skeie, O. Lysne, J. Flich, P. López, A. Robles, and J. Duato. „LASH-TOR: A Generic Transition-Oriented Routing Algorithm“. In: ICPADS '04: Proceedings of the Parallel and Distributed Systems, Tenth International Conference. Washington, DC, USA: IEEE Computer Society, 2004, p. 595.

[Toueg, 1980] S. Toueg. „Deadlock- and livelock-free packet switching networks“. In: STOC '80: Proceedings of the 12th annual ACM Symposium on Theory of Computing. New York, NY, USA: ACM, 1980, pp. 94–99.

References

[Varga, 2008] A. Varga and R. Hornig. „An overview of the OMNeT++ simulation environment“. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops. Simutools ’08. ICST, Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 60:1–60:10.

[Verma, 2010] A. Verma, S. Ajit, and D. Karanki. Reliability and Safety Engineering. Springer Series in Reliability Engineering. Springer, 2010.

[Wang, 2013] R. Wang, L. Chen, and T. M. Pinkston. „Bubble Coloring: Avoiding Routing- and Protocol-induced Deadlocks with Minimal Virtual Channel Requirement“. In: Proceedings of the 27th International ACM Conference on International Conference on Supercomputing. ICS ’13. New York, NY, USA: ACM, 2013, pp. 193–202.

[Yu, 2006] H. Yu, I.-H. Chung, and J. Moreira. „Topology Mapping for Blue Gene/L Supercomputer“. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. SC ’06. New York, NY, USA: ACM, 2006.

[Zahavi , 2010] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang. „Optimized InfiniBand fat-tree routing for shift all-to-all communication patterns“. In: Concurr. Comput. : Pract. Exper. 22.2 (Feb. 2010), pp. 217–231.

[F1] http://museum.ipsj.or.jp/en/computer/super/0020.html

[F2] http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_506_Spring_2010/ch_12_PP

[F3] https://asc.llnl.gov/computing_resources/bluegenel/

[F4] https://asc.llnl.gov/computing_resources/bluegenel/configuration.html

[F5] http://www.fujitsu.com/global/about/resources/news/press-releases/2011/0620-02.html

[F6] http://www.fujitsu.com/downloads/TC/sc10/interconnect-of-k-computer.pdf

[F7] http://www.netlib.org/utk/people/JackDongarra/PAPERS/tianhe-2-dongarra-report.pdf

[F8] http://www.netlib.org/utk/people/JackDongarra/PAPERS/tianhe-2-dongarra-report.pdf

[F9] https://www.top500.org/news/china-tops-supercomputer-rankings-with-new-93-petaflop-machine/

[F10] https://commons.wikimedia.org/wiki/File:General_architecture_of_the_Sunway_TaihuLight_system.png

[F11] http://www.sustainablecitiescollective.com/david-thorpe/198191/europes-most-congested-cities-and-how-cut-traffic-jams

[F12] https://commons.wikimedia.org/wiki/File:Autobahnen_in_Deutschland.svg

[F13-F15] http://domke.gitlab.io/paper/slides-domke-routing-2016.pdf

[F16] https://en.wikipedia.org/wiki/File:Kuniyoshi_Taiba_(The_End).jpg

[F17] https://pixabay.com/en/question-mark-punctuation-symbol-606955/