TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)

Routing on the Dependency Graph: A New Approach to Deadlock-Free High-Performance Routing

Jens Domke	TU Dresden
Torsten Hoefler	ETH Zurich
Satoshi Matsuoka	Tokyo Tech

Dipl.-Math. Jens Domke

Research Associate – Technische Universität Dresden
Institute of Computer Engineering – Computer Architecture

Email: jens.domke@tu-dresden.de
Tel.: +49 351 - 463 – 38783

ZIH
Center for Information Services & High Performance Computing

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**

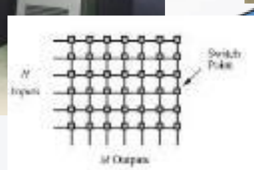
Motivation – Interconnection Networks for HPC-Systems

Towards ExaScale

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

1993: NWT (NAL)
140 Nodes
Crossbar Network

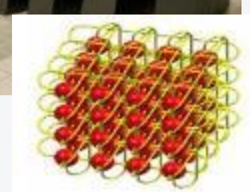
[F1]



[F2]

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

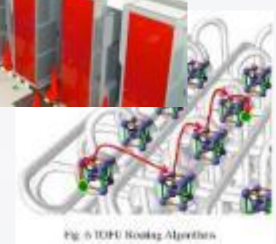
[F3]



[F4]

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

[F5]



[F6]

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

[F7]

[F8]

Routing Metrics:

- Low latency
- High throughput
- Low congestion
- Fault-tolerant
- Deadlock-free
- Low runtimes for fault recovery

Massive networks needed to connect all compute nodes of supercomputers (TOP500 [WEB, 2015])

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, terminals (N) and 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 DL-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

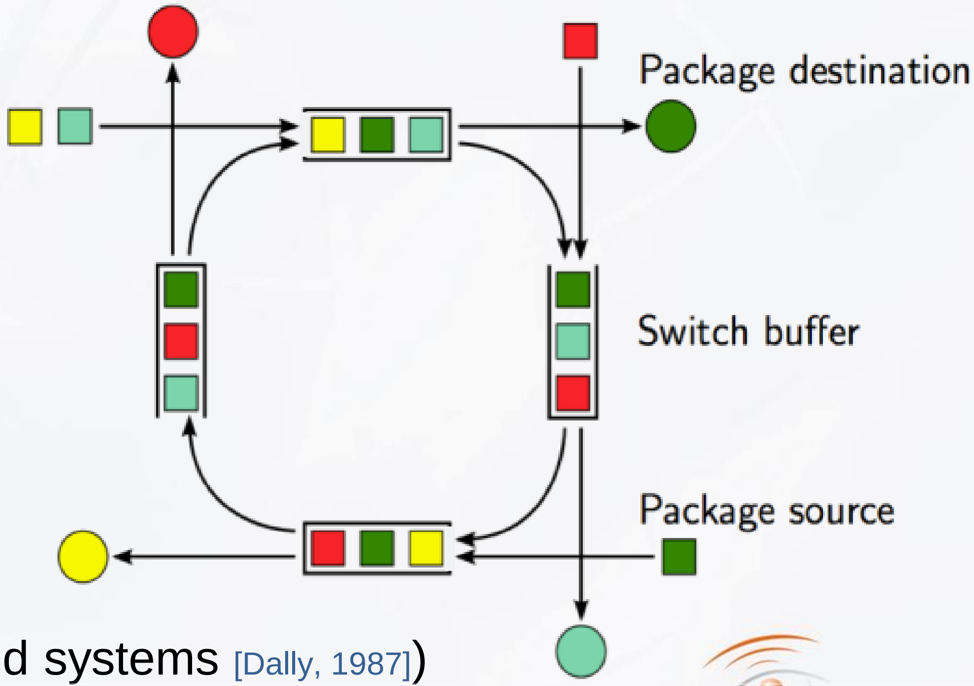
Routing Deadlocks – Credit Buffers in Lossless Interconnects

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])

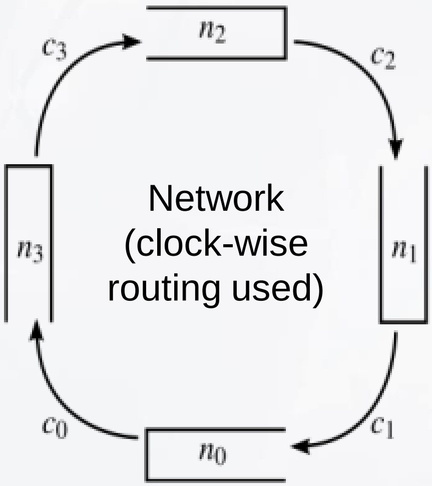
Theorem of Dally and Seitz [Dally, 1987]

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

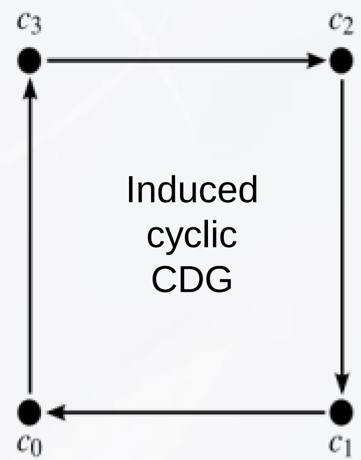
Channel Dependency Graph (CDG)

- Channels/links of $I := G(N,C)$ are nodes in the CDG $D := G(C,E)$, with ordered pairs $(n_x, n_y) =: c_i \in C$
- Connect nodes of C of the CDG if links are used to route messages

$$(c_i, c_j) \in E \Leftrightarrow \exists path := (\ldots, c_i, c_j, \ldots)$$



➔



Routing Deadlocks – Ignoring, Preventing, Avoiding, …

Ignoring routing deadlocks:

- ☹ “Resolving” via package life-time
- ☺ Fast path calculation (e.g., MinHop [Conte, 2002], SSSP [Hoefler, 2009])

Deadlock-prevention (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 (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., currently 8 and max. of 15 in IB [Shanley, 2003])

Others approaches, e.g.:

- Bubble Routing [Wang, 2013] ➔ not supported by current devices
- Controller principle [Toueg, 1980] ➔ global or local observer manages allocation of resources (doesn’t scale or currently not supported)

Routing Deadlocks – Virtual Channels or Virtual Networks

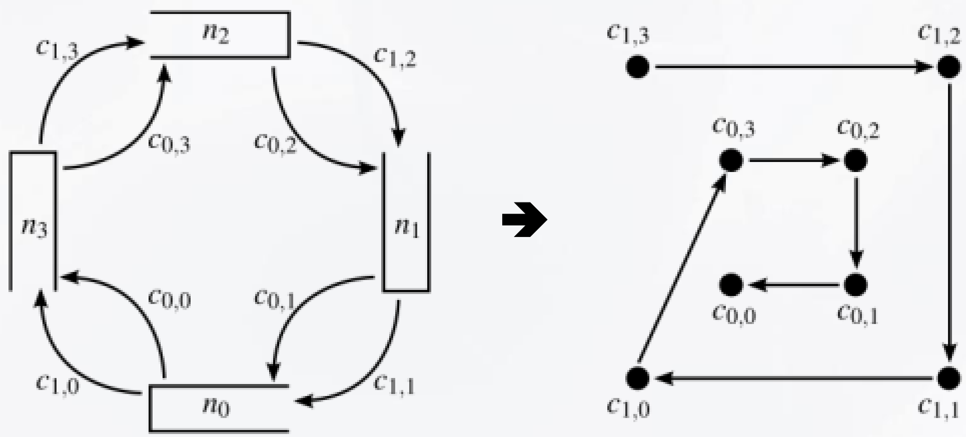
Virtual Channels

- Multiple sets of credit buffers in one port (all managed individually) [Dally, 2003]
- Split channels/links into multiple virtual channels

➡ Use different channels to generate acyclic CDG

VCs for deadlock-freedom (option 1)

- Use virtual channel transitioning to build acyclic CDG [Dally, 1987]
  (e.g., packets can switch between ‘high’ and ‘low’ channel)

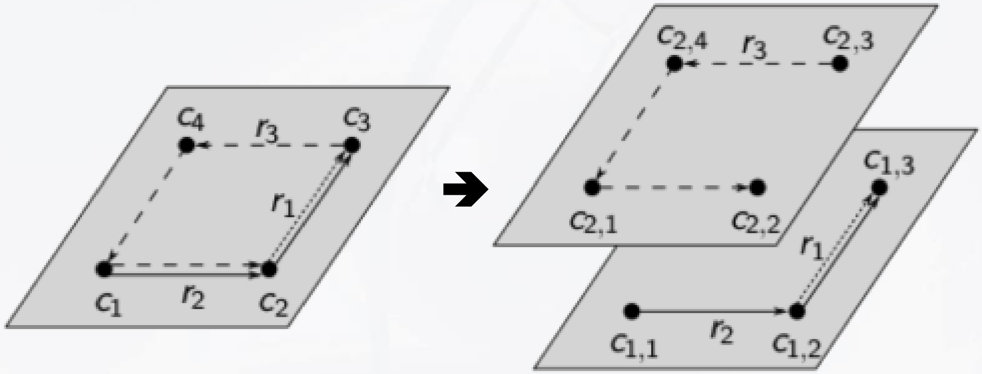


Routing Deadlocks – Virtual Channels or Virtual Networks

VCs for deadlock-freedom (option 2)

- Combine VCs into virtual layers [Skeie, 2002]
  (e.g., ‘high’ channels build ‘high’ layer and packets stay within one layer)
- Virtual layers == virtual networks and routes within a layer form acyclic CDG

➥ each layer is deadlock-free ➔ routing is deadlock-free



☹ VCs are limited due to implementation costs
(control logic, physical buffer size, etc.)

Related Work: Comparison of existing Routing Algorithms

Routing	Network $I=G(N,C)$	Latency	Through-put	Deadlock-Freedom	VC	Fault-Tolerant	Time Complexity[♯]
DOR [Rauber, 2010]	meshes	+	+	yes	1	no	N/A
Torus-2QoS [MLX, 2003]	2D/3D meshes/tori	+	++	yes	≥ 2	limited	N/A
Fat-Tree [Zahavi, 2010]	k-ary n-tree	+	++	yes	1	limited	N/A
MinHop [Conte, 2002]	arbitrary	+	+	no	1	yes	$O(\|N\| \bullet \|C\|)$
Up/Dn [Schroeder, 1991]	arbitrary	- -	- -	yes	1	yes	$O(\|N\| \bullet \|C\|)$
MUD [Flich, 2002]	arbitrary**	-	-	yes	≥ 2	yes	$O(\|N\| \bullet \|C\|)$
(DF)SSSP [Domke,'11;Hoefler,'09]	arbitrary	+	++	(yes*) no	(≥)1	yes	$O(\|N\|^2 \bullet log\|N\|)$
LTURN [Koibuchi,'01]	arbitrary	-	-	yes	1	yes	$O(\|N\|^3)$
LASH [Skeie, 2002]	arbitrary	+	-	yes*	≥ 1	yes	$O(\|N\|^3)$
LASH-TOR [Skeie,'04]	arbitrary**	-	-	yes	≥ 1	yes	$O(\|N\|^3)$
SR [Mejia, 2006]	arbitrary	-	-	yes	1	yes	$O(\|N\|^3)$
Smart [Cherkasova,'96]	arbitrary	-	+	yes	1	yes	$O(\|N\|^9)$

♯: to (re-)calculate all LFTs for network I [Flich, 2012]

*: limited; might exceed available #VCs

**: not easily applicable for destination-based forwarding

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

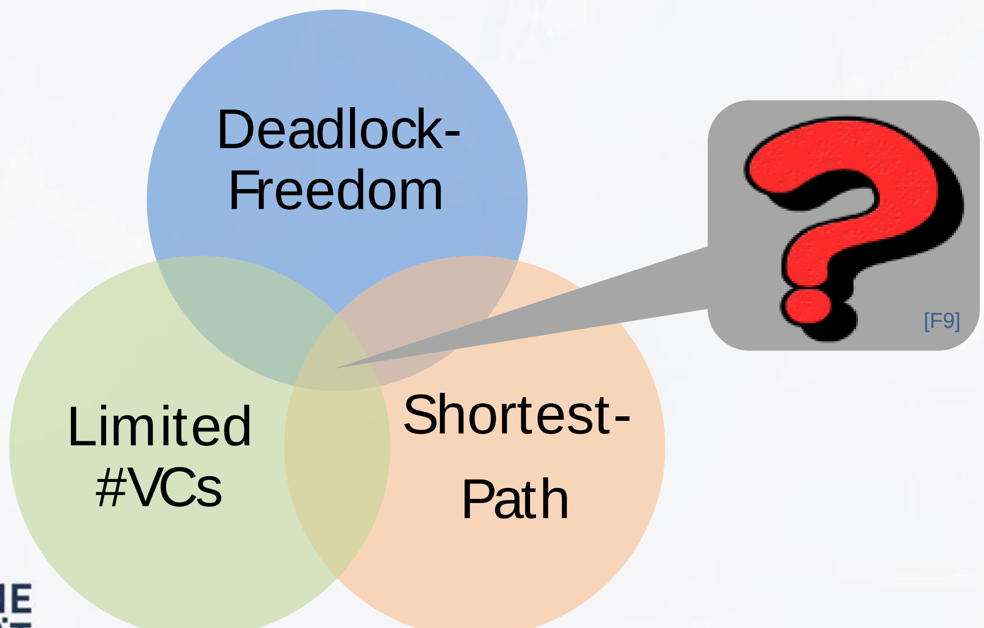
Routing Deadlocks – Deadlock-Freedom and Shortest-Path

Assumptions:

- Arbitrary topology
- Arbitrary but fixed number of VCs (0/1, 2, or more…)
- Destination-based routing algorithm

Question:

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

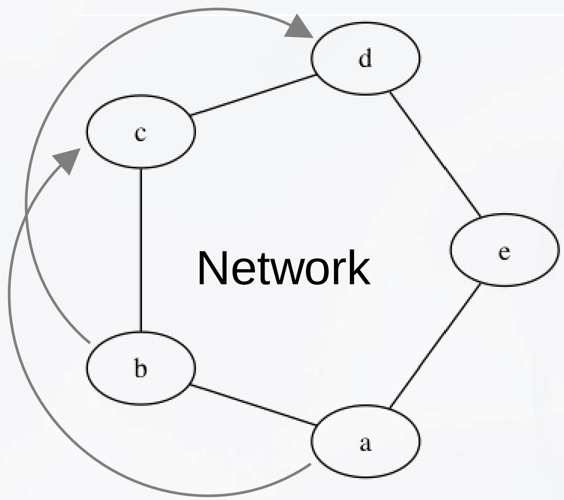


Routing Deadlocks – Deadlock-Freedom and Shortest-Path

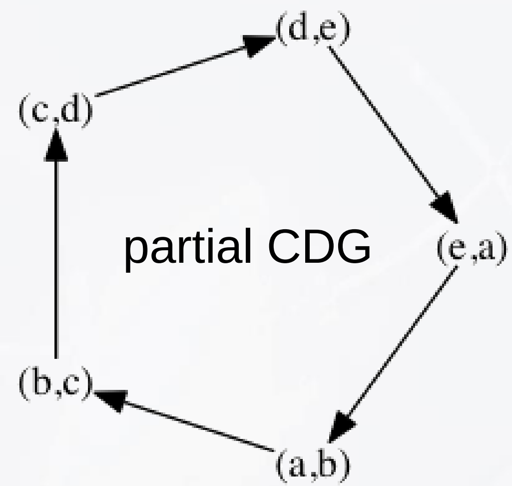
Easy **counter example**, assume:

- Ring network with 5 nodes; no/one virtual channels; shortest-path routing
- Node ***a*** sends messages to ***c***; ***b*** sends to ***d***; ***c*** sends to ***e***; …

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



➔



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

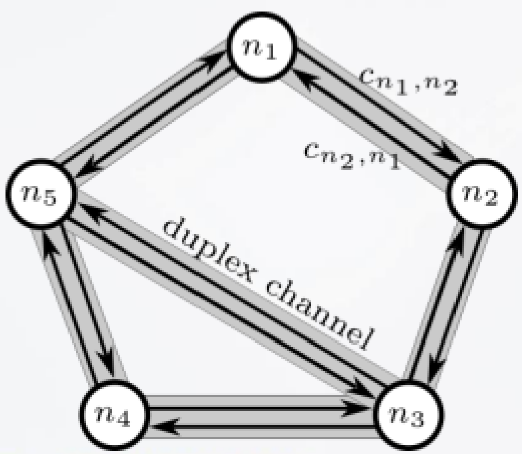
Step 1: restriction of possible turns
Step 2: calculate (non-shortest) paths
➥ ☹ overly restrictive; poor balancing

Virtual Channel Approach

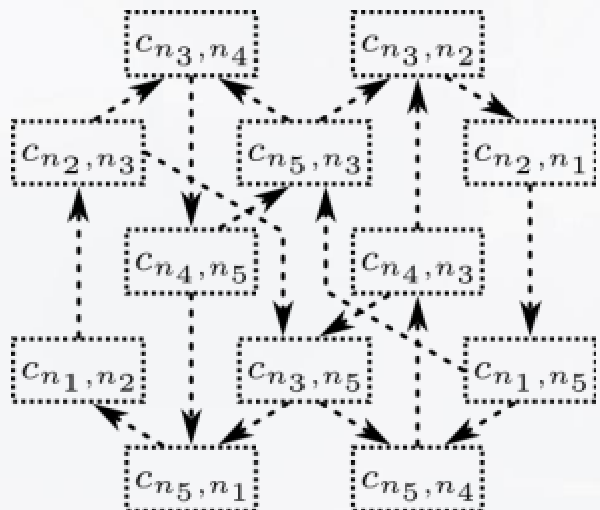
Step 1: calculate shortest paths in *I*
Step 2: create acyclic *CDGs* per VL
➥ ☹ needed #VCs is unbound

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

Network *I* (ring w/ shortcut)



➔



Complete Channel Dependency Graph

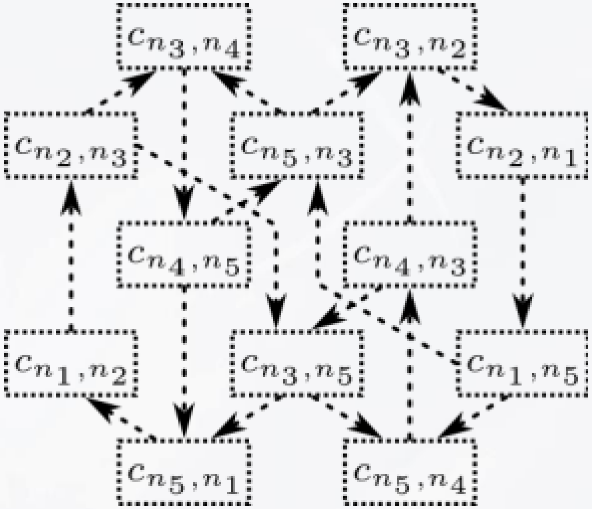
Complete Channel Dependency Graph

What is the **complete CDG**?

$$\overline{D} := G(C, \overline{E})$$, with

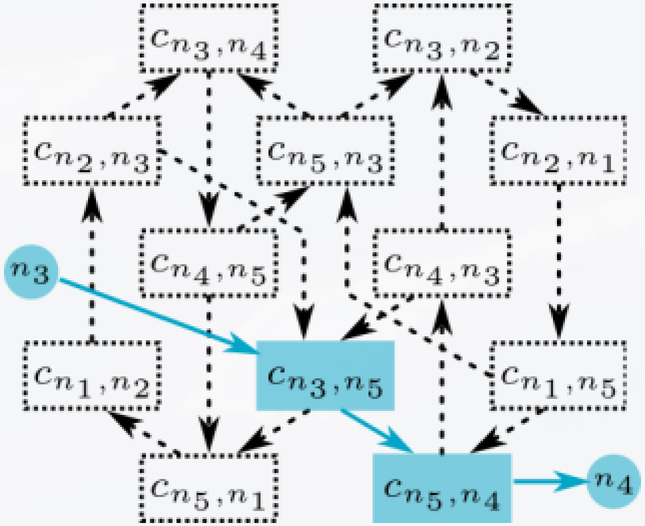
$$\forall (n_x, n_y), (n_y, n_z) \in C, n_x \neq n_z : ((n_x, n_y), (n_y, n_z)) \in \overline{E}$$

- Includes node/link information
- Includes all possible routes
  (i.e., all available channel dependencies)
- Size of $\overline{D}$:
  - $|C| = 2 \cdot |\#\{links\ of\ I\}|$
  - $|\overline{E}| \leq (\max(switch\ \ radix) - 1) \cdot |C|$
- Initially: all edges $\in \overline{E}$ are in ***unused*** state



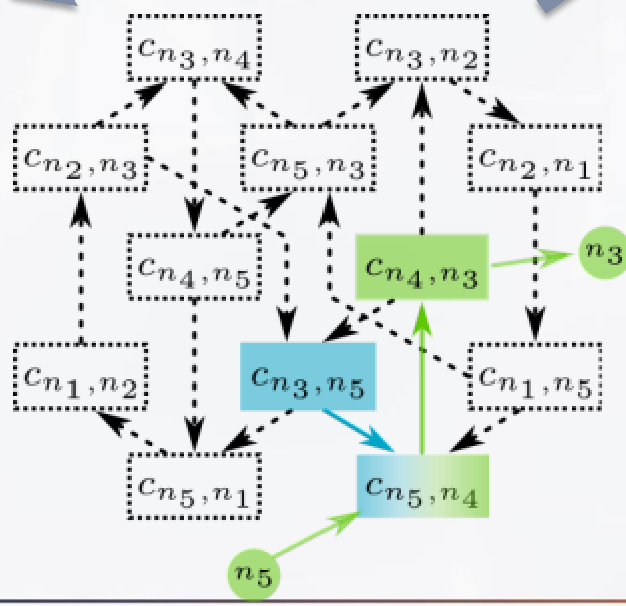
➥ Allows “on-demand” checks for acyclic subgraphs ☺

Routes in the Complete Channel Dependency Graph



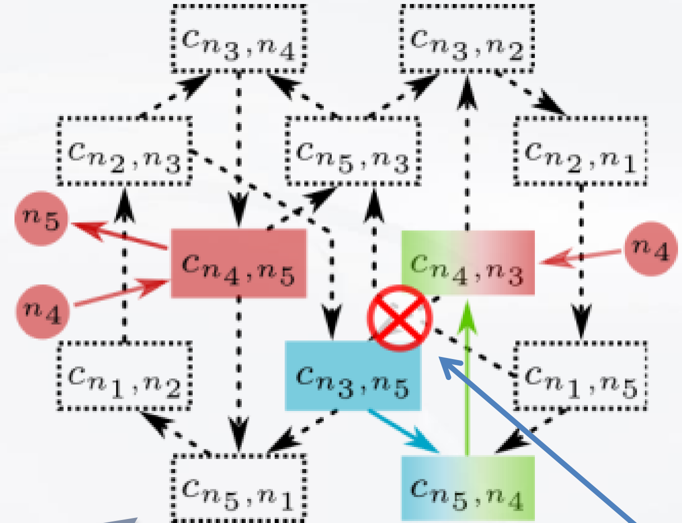
Step 1

- Route from n_3 to n_4 via node n_5
- Change edge between $c_{n3,n5} \rightarrow c_{n5,n4}$ from ***unused*** state into new ***used*** state



Step 2

- Route from n_5 to n_3 via n_4
- Change edge to ***used*** state



Step 3

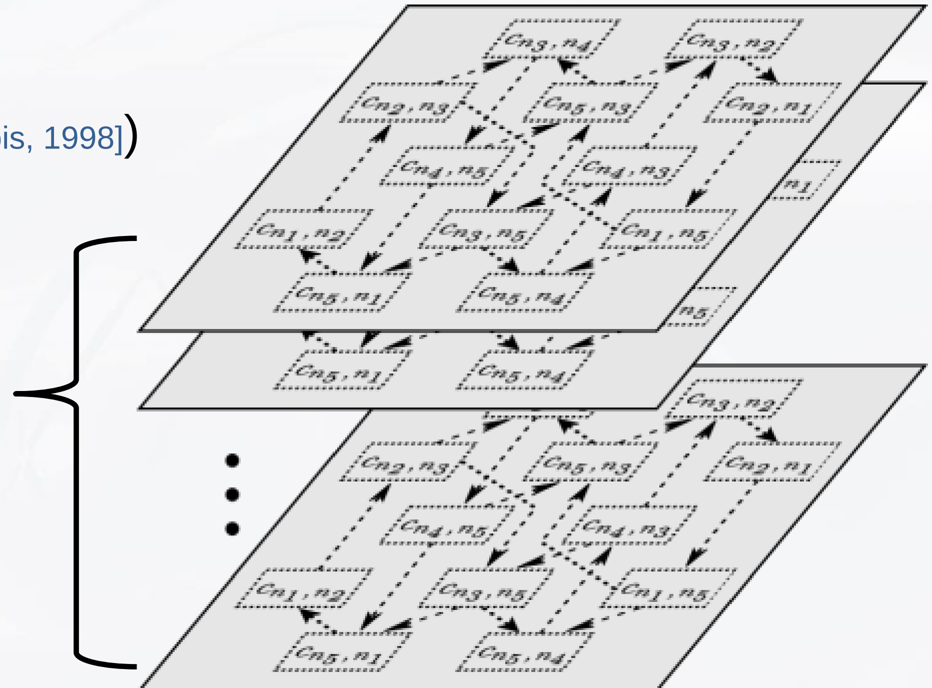
- Route from n_4 to n_5 via n_3?
  ➡ closes cycle in $\overline{D}$
  ➡ mark edge ***blocked***
- Use alternative (direct route) given by $c_{n4,n5}$

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



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

- via modified Dijkstra's algorithm on complete CDG $\overline{D}$ (similar to (DF)SSSP routing on I)
- Destination $n_d \in N_i^d$ acts as source node for Algorithm 1
- Main difference: use edge if and only if no cycle is created

Path balancing

- Use weights for channels (additionally to node distances)
- Update channel weights of used links after Algo. 1 finished

➥ Minimizes overlapping of routes if possible

Algorithm 1: 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)

```
foreach node n ∈ N do
    n.distance ← ∞
    n.usedChannel ← ∅
n_0.distance ← 0
c_0.distance ← 0
FibonacciHeap Q ← {c_0}
while Q ≠ ∅ do
    c_p ← Q.findMin()
    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
        if c_p.distance + c_q.weight < n_{c_q}.distance then
            (c_p, c_q).state ← used                    // modifies D̄
            if D̄ is cycle-free then
                Q.add(c_q)
                c_q.distance ← c_p.distance + c_q.weight
                n_{c_q}.distance ← c_p.distance + c_q.weight
                n_{c_q}.usedChannel ← c_q
            else
                (c_p, c_q).state ← blocked
```

Checking for Absence of Cycles in the Complete CDG

Do we have to check every edge?

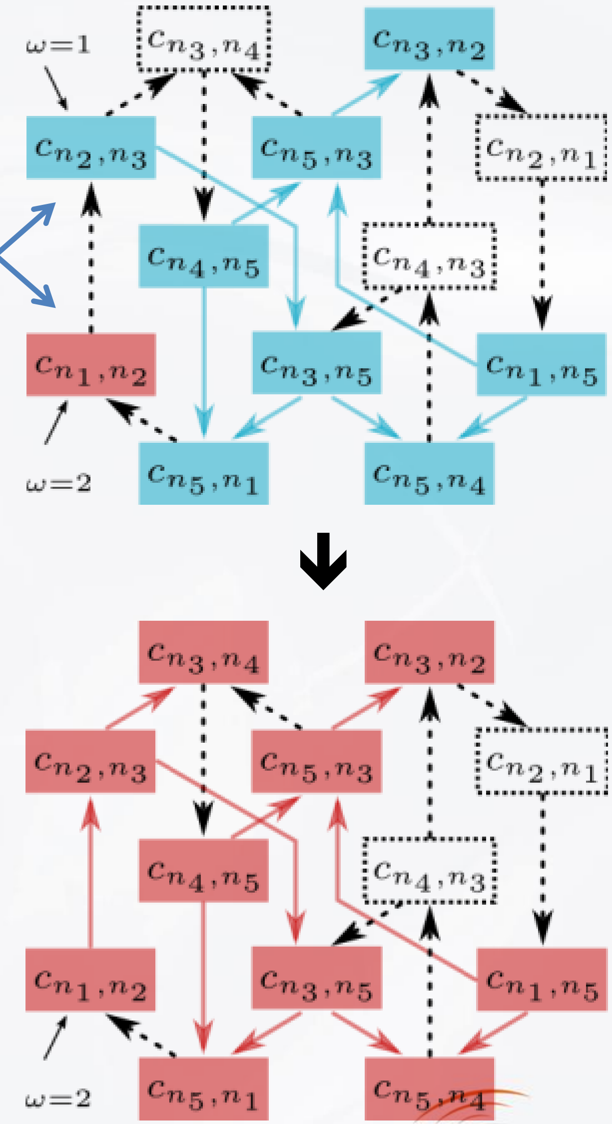
- New subgraph identification (ω) for each call to Dijkstra's (prev. slide)
- ω gets assigned to each node/edge of $\overline{D}$ identifying connected/acyclic subgraphs

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

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

➥ Cycle check for edge e needed?

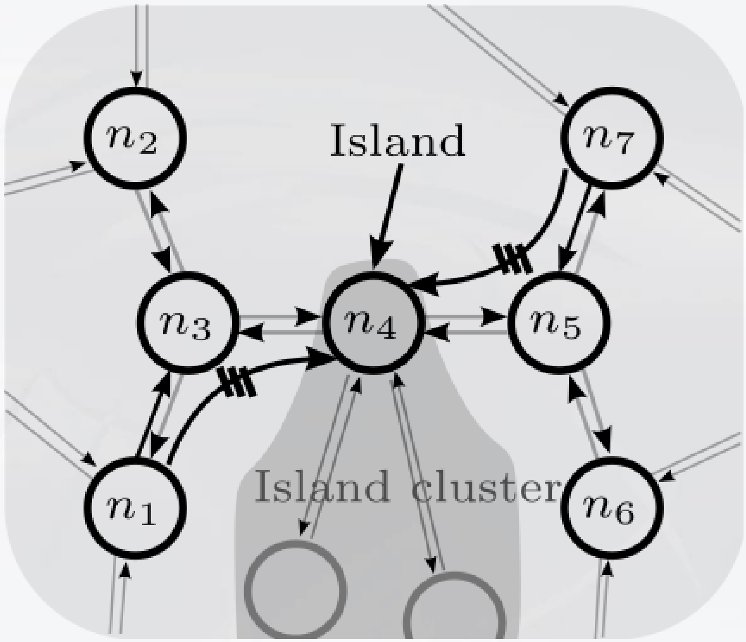
- No
  - $\omega(e) = -1$, already ***blocked***
  - $\omega(e) \geq 1$, already ***used***
  - merging two different acyclic subgraphs ➔ acyclic again
- Yes
  - $\omega(e) = 0$ and same ω for adjacent nodes



Routing Impasse and Fallback to Escape Paths

Problems

- Iterative path calculation within D can get stuck
  ➥ not all nodes discoverable



Possible solutions

- Backtracking (similar to 8-queens problem, #q >> 8) ➔ very expensive ☹
- 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

➥ very time- and memory efficient
➥ local backtracking works for most impasses

Pseudo Code of Nue Routing

Algorithm 2: 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 `// Section 4.3`
    Identify central $n_{r,i} \in N_i^H$ of H_i `// Section 4.3`
    Create a new complete CDG $\overline{D}_i$ `// Section 4.1`
    Define escape paths D_i^s for root $n_{r,i}$ `// Section 4.2`
    foreach *Node* $n \in N_i^d$ **do**
        Identify deadlock-free paths $P_{\cdot,n}$ `// Section 4.4`
        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

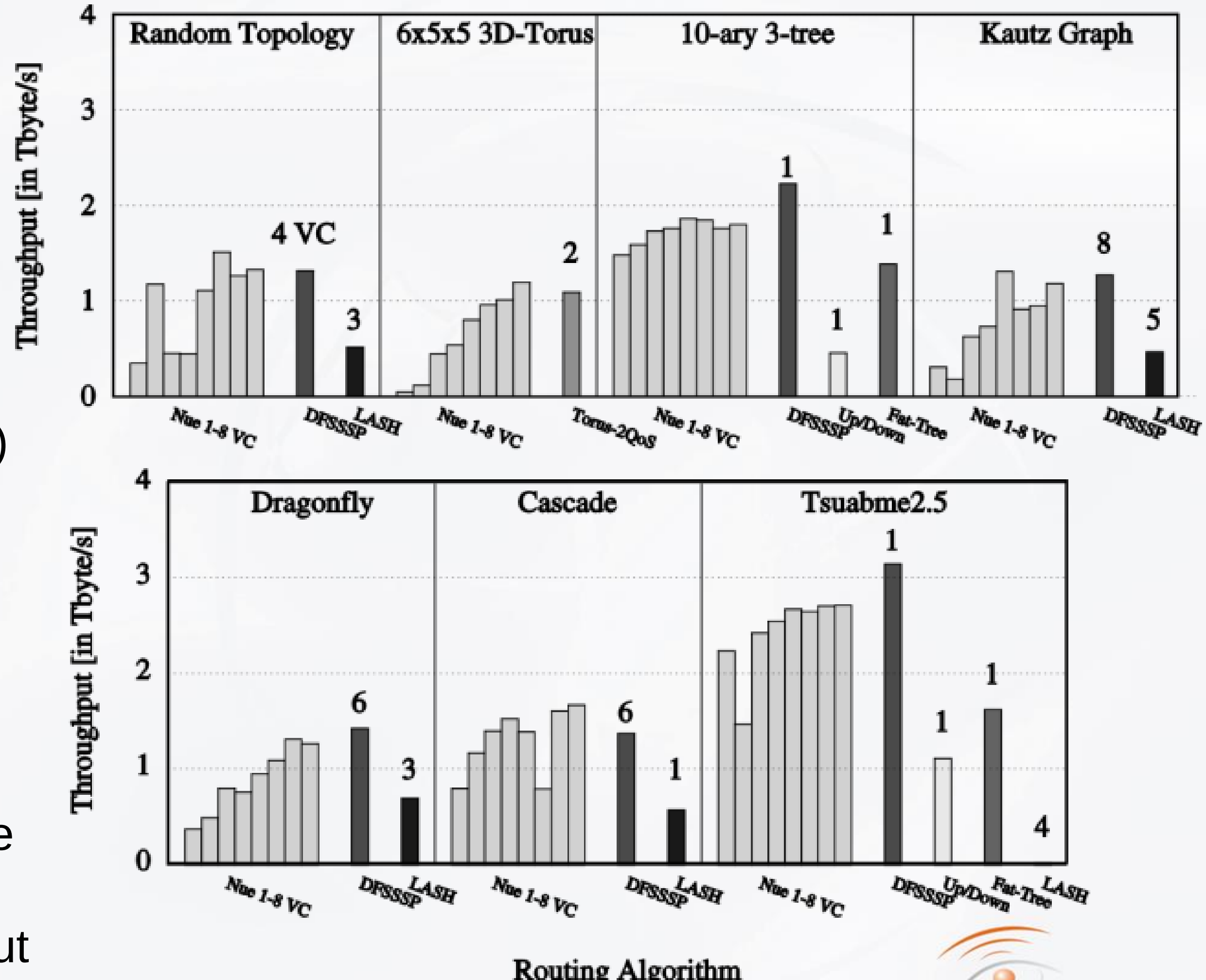
- Flit-level simulation framework for IB (OMNet++ [Varga, 2008] & ibmodel [Gran, 2011])
- Communication throughput of all-to-all traffic pattern (similar to MPI_Alltoall) with 2 KiB messages
- Multiple topologies with approx. 1,000 compute nodes (or terminals)
- Comparison of Nue to all routing algorithms 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; and integrated in simulation framework for fair runtime comparison
- Created 25 3D torus networks (size: 2x2x2, 2x2x3, 2x3x3,…, 10x10x10) with 4 terminal nodes per switch; 4xQDR IB with 8 VCs
- 1% randomly inject link/channel failures (common annual failure rate [Domke, 2014])

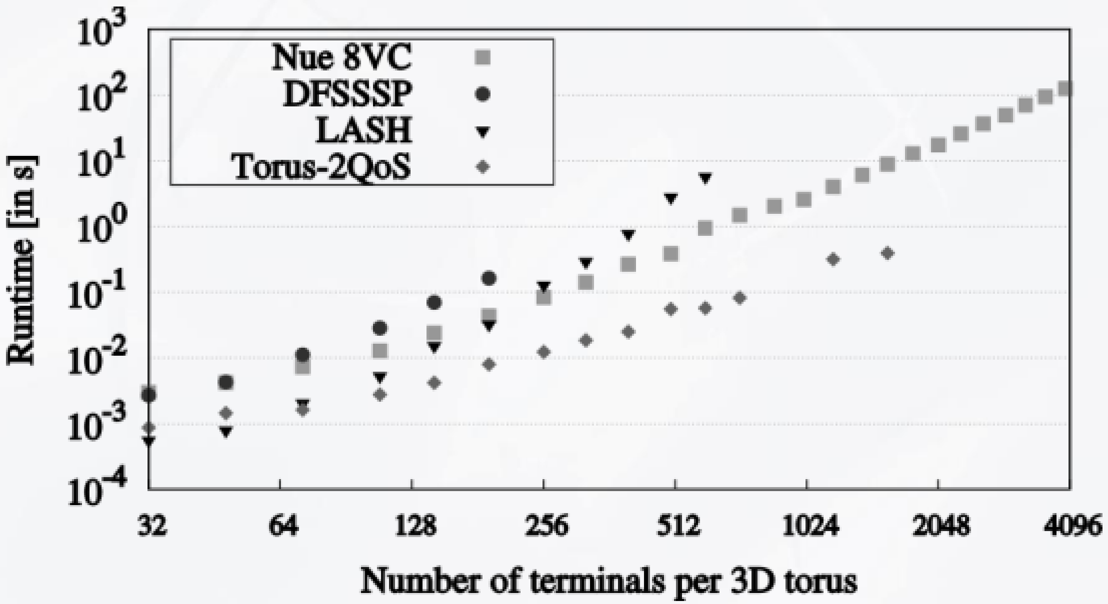
Result

☹ DFSSSP/LASH run out of VCs (➔ not deadlock-free)

☹ Torus-2QoS not fault-tolerant enough

☺ Nue is always applicable

☺ Faster routing calculation with Nue vs. DFSSSP/LASH (at larger scale)



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	Throughput	Deadlock-Freedom	VC	Fault-Tolerant	Time Complexity[♯]
DOR	meshes	+	+	yes	1	no	N/A
Torus-2QoS	2D/3D meshes/tori	+	+ +	yes	≥ 2	limited	N/A
Fat-Tree	k-ary n-tree	+	+ +	yes	1	limited	N/A
(DF)SSSP	arbitrary	+	+ +	(yes*) no	(≥)1	yes	$O(\|N\|^2 \bullet log\|N\|)$
• • •							
LASH	arbitrary	+	-	yes*	≥ 1	yes	$O(\|N\|^3)$
LASH-TOR	arbitrary* *	-	-	yes	≥ 1	yes	$O(\|N\|^3)$
SR	arbitrary	-	-	yes	1	yes	$O(\|N\|^3)$
Smart	arbitrary	-	+	yes	1	yes	$O(\|N\|^9)$
Nue	**arbitrary**	**+**	**+ / + +**	**yes**	**≥ 1**	**yes**	$O(\|N\|^2 \bullet log\|N\|)$

♯: to (re-)calculate all LFTs for network I

* : limited; might exceed available #VCs

* * : not easily applicable for destination-based forwarding

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Conclusions

- Future (and current) networks will be:
  - Lossless (see RoCE(v2) [Zhu, 2015; IB-A17, 2014], Intel Omni-Path [Birrittella, 2015], InfiniBand [Shanley, 2003], …)
  - Much bigger, but sparse or irregular (e.g., fail-in-place networks [Domke, 2014])
- Oblivious, destination-based Nue routing for HPC:
  - 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
    - ➥ Combining Quality-of-Service (QoS) and deadlock-freedom for IB
  - Offers competitive bandwidth/latency and path calculation time
  - Applicable to statically routed technologies (e.g., IB, OPA, RoCE, …)
  - Nue routing for escape paths ($\boldsymbol{R_1}$) of fully adaptive routing (see Duato's protocol [Dally, 2003])

Thank you for your attention!

Nue – Japanese chimera combining the advantages of existing routing algorithms

Nue routing for InfiniBand (OpenSM implementation):
http://spcl.inf.ethz.ch/Research/Scalable_Networking/Nue/

[F10]

References (A-D)

[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. Underwood, and R. C. Zak, "Intel® Omni-path Architecture: Enabling Scalable, High Performance Fabrics," in High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on, 2015, pp. 1-9.

[Cherkasova, 1996] L. Cherkasova, V. Kotov, and T. Rokicki, "Fibre channel fabrics: evaluation and design," in System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on , 1996, vol. 1, pp. 53-62 vol.1.

[Coffman, 1971] E. G. Coffman, M. Elphick, and A. Shoshani, "System Deadlocks," ACM Computing Surveys, vol. 3, no. 2, pp. 67-78, 1971.

[Conte, 2002] M. Conte, Dynamic Routing in Broadband Networks. Springer US, 2002.

[Dally, 1987] 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.

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

[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, 2011, pp. 613-624.

[Domke, 2014] J. Domke, T. Hoefler, and S. Matsuoka, "Fail-in-Place Network Design: Interaction between Topology, Routing Algorithm and Failures," in Proceedings of the IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC14), New Orleans, LA, USA, 2014, pp. 597-608.

References (E-K)

[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, London, UK, UK, 2002, pp. 49-63.

[Flich, 2012] 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.

[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.

[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), 2009.

[IB-A17, 2014] "Supplement to InfiniBandTM Arch. Spec. Volume 1 Release 1.2.1 (Annex A17: RoCEv2).“ Sep-2014.

[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.

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

[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.

[Kung, 1994] H. T. Kung, T. Blackwell, and A. Chapman, "Credit-based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation and Statistical Multiplexing," in Proceedings of the Conference on Communications Architectures, Protocols and Applications, New York, NY, USA, 1994, pp. 101-114.

[Karypis, 1998] G. Karypis and V. Kumar. “Multilevel K-way Partitioning Scheme for Irregular Graphs,” in J. Parallel Distrib. Comput., 48(1):96–129, 1998.

References (L-T)

[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 Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, 2006, p. 10 pp.-.

[MLX, 2013] "Mellanox OFED for Linux User Manual Rev. 2.0-3.0.0." Mellanox Technologies, 2013.

[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," IEEE Journal on Selected Areas in Communications, vol. 9, no. 8, Oct. 1991.

[Shanley, 2003] T. Shanley, J. Winkles, and I. MindShare, InfiniBand Network Architecture. 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, 2002, p. 194.

[Skeie, 2004] T. Skeie, O. Lysne, J. Flich, P. Lopez, 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, 2004, p. 595.

[Toueg, 1980] S. Toueg, "Deadlock- and livelock-free packet switching networks," in STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing, New York, NY, USA, 1980, pp. 94-99.

References (U-Z)

[Varga, 2008] 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.

[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, New York, NY, USA, 2013, pp. 193-202.

[WEB, 2015] Prometeus GmbH, "TOP500," 2015. [Online]. Available: http://www.top500.org/.

[Zahavi, 2010] 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.

[Zhu, 2015] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion Control for Large-Scale RDMA Deployments," in Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, New York, NY, USA, 2015, pp. 523-536.

Figure References (1-9)

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

[F10] https://en.wikipedia.org/wiki/File:Kuniyoshi_Taiba_(The_End).jpg