



Deadlock-Free Oblivious Routing for Arbitrary Topologies

Jens Domke, Torsten Hoefler, Wolfgang E. Nagel

May 18th, 2011

Zellescher Weg 12
Willers-Bau A 219
01062 Dresden
Tel. +49 0351 - 463 39114

Jens Domke (jens.domke@tu-dresden.de)

Outline

- 1 Basics and previous work
- 2 Deadlocks
- 3 Deadlock-free SSSP routing algorithm
- 4 Simulations and measurements
- 5 Conclusion

Basics and previous work

- InfiniBand interconnect
- InfiniBand subnet manager – OpenSM
- Motivation
- Previous work

InfiniBand interconnect

- Based on an open standard, developed by the InfiniBand Trade Association
- One of the most widely used interconnect in the field of HPC

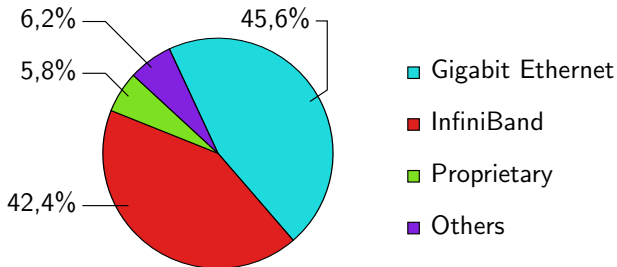


Figure: Top500 List, Interconnects, Nov. 2010

Tasks

- Scan the components of the IB subnet
- Initialize the IB ports
 - Calculate paths for each port pair in the subnet
 - Generate linear forwarding tables (LFT)
 - Configure the IB ports with additional preferences, e.g. QoS
- Reconfiguration, if the subnet changes

Implemented static/destination-based routing algorithms

- MinHop
- Up*/Down*
- Fat-Tree
- LASH
- DOR

General problems for most of the routing algorithms

- No global balancing of the traffic \Rightarrow congestions reduce the bandwidth
- Only designed for a small set of topologies
- Not deadlock-free for every topology
- Not usable for production systems, because of long runtime

The algorithm should support irregular topologies, because

- HPC-systems grow in their lifetime
- Additional node like I/O or login nodes are connected
- Network components can fail

Single-source-shortest-path routing algorithm

- "Optimized Routing for Large-Scale InfiniBand Networks" [Hoefler et al., 2009] presented SSSP
- Minimizes congestions thru global balancing
- Higher effective bisection bandwidth compared to others algorithms
- Disadvantage of the presented approach
 - Algorithm is not deadlock-free
 - LFT are calculated by an external program (not OpenSM)

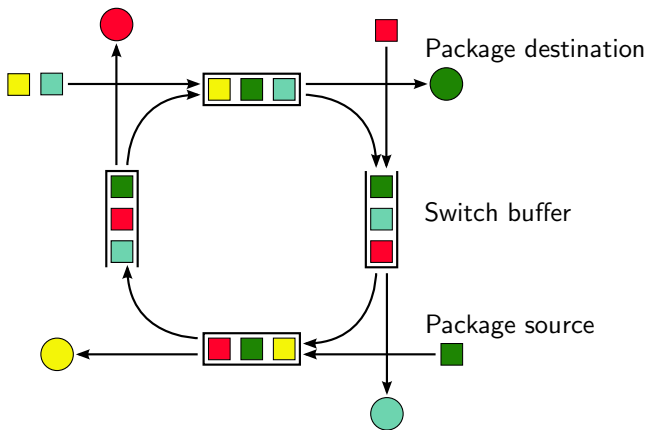
Deadlocks

- Definition
- Deadlocks in interconnects
- Approaches for deadlock-free routing
- Theorem of Dally and Seitz
- Virtual channels and channel dependency graph

Definition Deadlock [Tanenbaum, 2007]

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

Deadlocks in interconnects



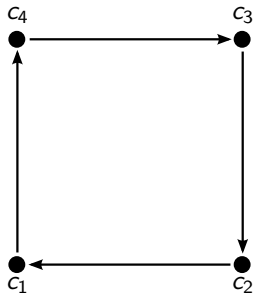
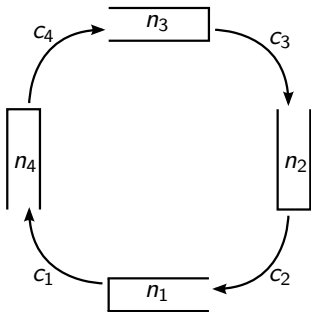
Approaches for deadlock-free routing

- Package life-time (only to break the deadlock, if they occur)
- Controller principle
- Up*/Down* routing
- Virtual channels
 - "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks" [Dally and Seitz, 1987]
 - Each link will be split into multiple virtual channels
 - Channel dependency graph

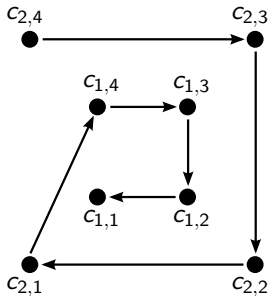
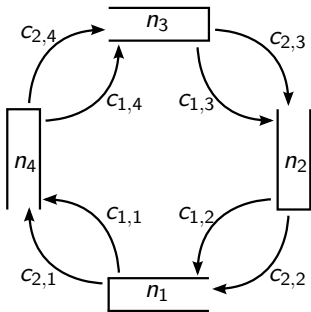
Theorem of Dally and Seitz

A routing algorithm for a interconnect is deadlock-free, iff there are no cycles in the corresponding channel dependency graph.

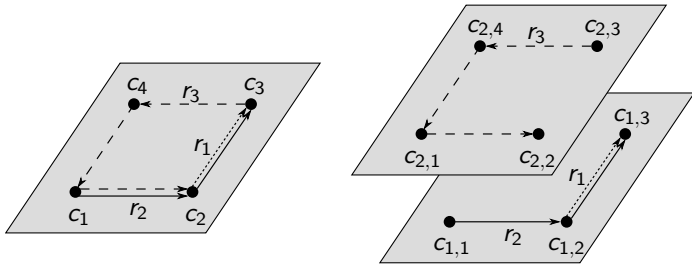
Virtual channels and channel dependency graph



Virtual channels and channel dependency graph



Virtual channels and channel dependency graph



Deadlock-free SSSP routing algorithm

- DFSSSP routing algorithm
- How to identify the "weakest" edge?

Algorithm 1 DFSSSP routing algorithm

/ Phase 1: Identification of all network components */*

Scan(...)

/ Phase 2: Calculate paths */*

SSSP(...)

/ Phase 3: Assign paths to virtual layers */*

RemoveDeadlocks(...)

/ Phase 4: Balancing of all virtual layers */*

Balancing(...)

DFSSSP routing algorithm

Algorithm 2 Remove deadlocks from the channel dependency graph (Phase 3)

Input: Linear forwarding tables

Output: Assign each path to a virtual layer

/ Initialization of layer 1 */*

for all PortPairs(source, destination) **do**

 Update CDG[1] with the source-destination path

end for

/ Search cycles in the channel dependency graph */*

for $i = 1, \dots, max - 1$ **do**

repeat

 Search for cycle in CDG[i]

 Identify "weakest" edge of the cycle

 Move port pairs or paths on this edge to CDG[$i + 1$]

until no cycle found in CDG[i]

end for

Search for cycle in CDG[max]

How to identify the "weakest" edge?

... to minimize the number of needed virtual layers.

Abstract formulation: "acyclic path partitioning" problem (APP)

- Split a set of paths into subsets which produces acyclic channel dependency graphs.
- Shown to be NP-complete
- Proof based on an polynomial transformation from graph k -colorability problem into APP

APP is NP-complete \Rightarrow use heuristic to identify the "weakest" edge

- Edge with most paths in the cycle
- Random edge of the cycle
- **Edge with smallest number of paths**

Simulations and measurements

- Simulations with IBSim
 - Real existing topologies
- Measurements on a real system – Deimos
 - PC-Farm Deimos
 - Netgauge
 - BenchIT
 - NAS parallel benchmarks

Real existing topologies

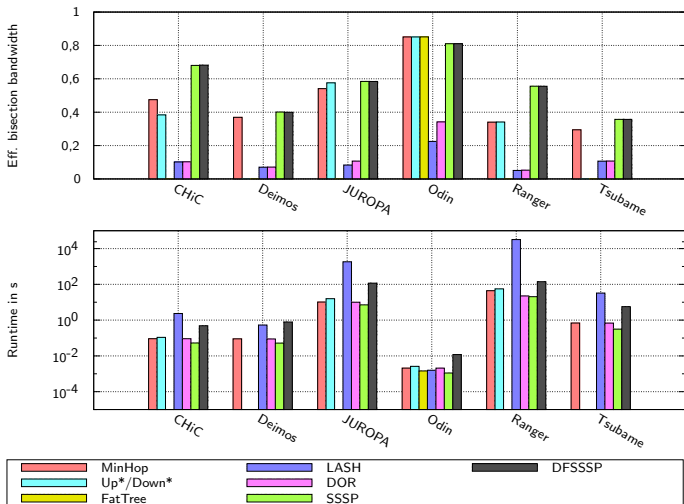


Figure: Simulation with IBSim and ORCS [Schneider et al., 2009]

Measurements on a real system – Deimos

HPC-system operated by ZIH

- Linux Network PC-Farm (13.9 TFlop/s)
- 726 compute nodes connected by 108 IB switches
- 2,6 GHz AMD Opteron X85 dual core
- 1, 2 or 4 processors per node
- 2 GByte RAM per core



Measurement environment and used benchmarks

- Exclusive access
- One MPI process per node (for measurements with ≤ 512 cores)
- Same number of MPI processes \implies same compute nodes used
- Eff. bisection bandwidth with Netgauge [Hoefler et al., 2007]
- Runtime and bandwidths of pure MPI communication measured with micro-benchmarks (BenchIT [Juckeland et al., 2004])
- Performance gain for application benchmarks of NASA (NAS Parallel Benchmarks [Bailey et al., 1995])

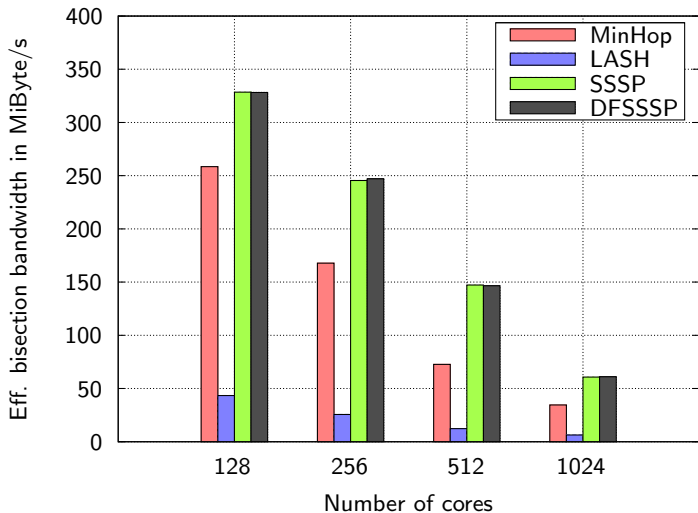


Figure: Approximation with 1000 random bisections

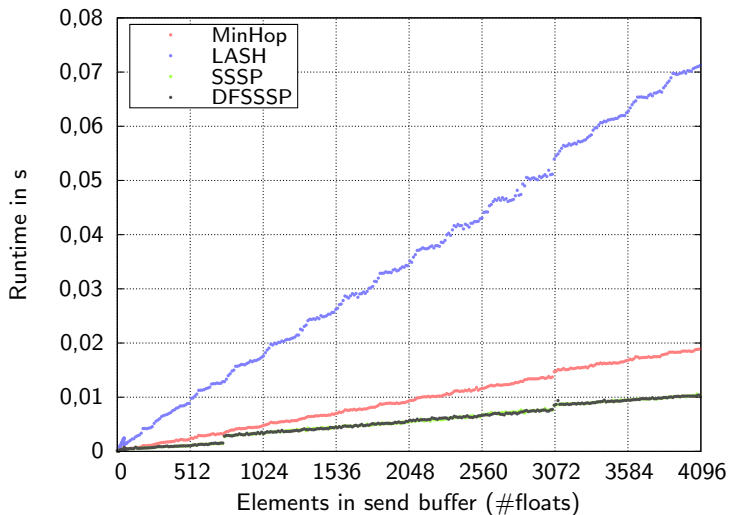


Figure: Collective N -to- N MPI operation on 128 nodes

NAS parallel benchmarks

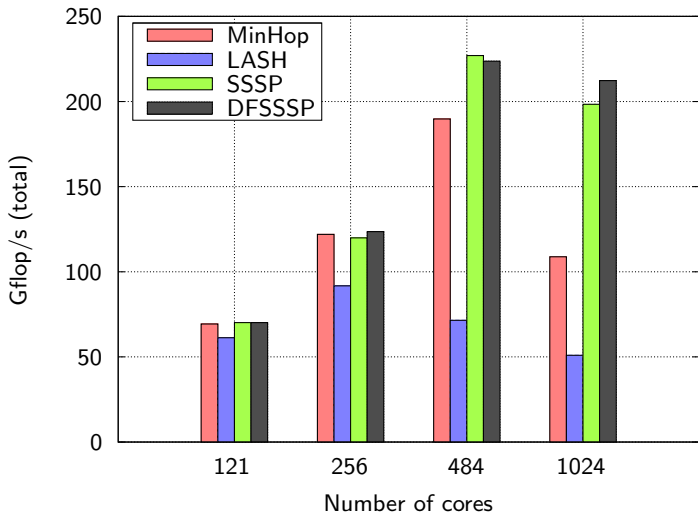


Figure: BT, class C – equation system solver

Conclusion

- Developed deadlock-free SSSP routing for arbitrary network topologies
- DF-/SSSP routing algorithm integrated in OpenSM
 - Patch available: <http://unixr.de/research/dfsssp/>
- Not limited to InfiniBand; usable for all interconnects which support virtual channels
- Modeled the "acyclic path partition" problem; proofed NP-completeness
- Doubled the eff. bisection bandwidth of Deimos for 512 nodes
- Performance gain (communication bound) for application benchmarks up to 95%

References

- D. Bailey, T. Harris, W. Saphir, R. V. D. Wijngaart, A. Woo, and M. Yarrow. The nas parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center, Dec. 1995.
- W. Dally and C. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *Computers, IEEE Transactions on*, C-36(5): 547–553, May 1987. ISSN 0018-9340. doi: 10.1109/TC.1987.1676939.
- T. Hamada and N. Nakasato. InfiniBand Trade Association, InfiniBand Architecture Specification, Volume 1, Release 1.0. In *International Conference on Field Programmable Logic and Applications*, pages 366–373, 2005.
- T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm. Netgauge: A Network Performance Measurement Framework. In *High Performance Computing and Communications, Third International Conference, HPCC 2007, Houston, USA, September 26-28, 2007, Proceedings*, volume 4782, pages 659–671. Springer, Sept. 2007. ISBN 978-3-540-75443-5.
- 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.
- G. Juckeland, S. Börner, M. Kluge, S. Kölling, W. Nagel, S. Pflüger, H. Röding, S. Seidl, T. William, and R. Wloch. Benchit – performance measurement and comparison for scientific applications. In F. P. G.R. Joubert, W.E. Nagel and W. Walter, editors, *Parallel Computing - Software Technology, Algorithms, Architectures and Applications*, volume 13 of *Advances in Parallel Computing*, pages 501–508. North-Holland, 2004.
- T. Schneider, T. Hoefler, and A. Lumsdaine. ORCS: An Oblivious Routing Congestion Simulator. Technical Report 675, Indiana University, Feb. 2009.
- A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3. edition, 2007. ISBN 9780136006633.

Time complexity

The time complexity for the DFSSSP routing algorithm is

$$O(|N|^2 \cdot (\log |N| + \nabla) + |N| \cdot |C| + \nabla \cdot (|C| + |E|))$$

Memory complexity

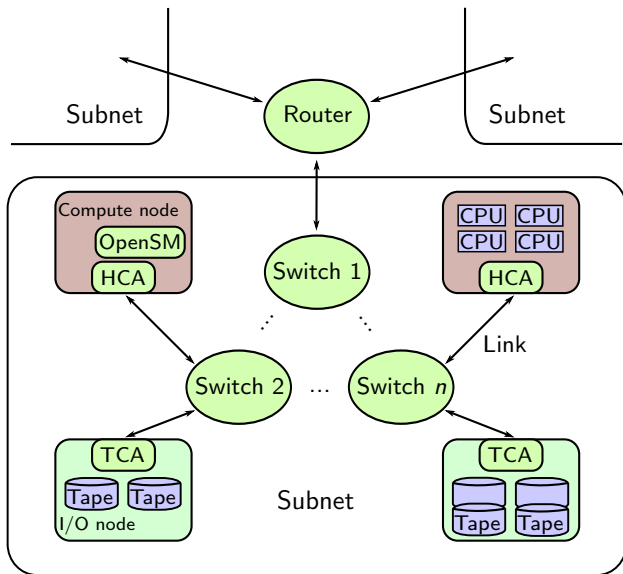
The memory complexity for DFSSSP is

$$O(\nabla \cdot d(l) \cdot |N|^2 + \nabla \cdot (|C| + |E|) + |N|)$$

Variables:

- N – nodes in the network
- C – channels/links
- E – edges in the channel dependency graph
- ∇ – minimal number of needed virtual layer
- $d(l)$ – diameter of network l

Backup – InfiniBand subnet



- Significant properties
 - Low latency
 - High bandwidth for package transfer
 - Absence of deadlocks in the routing
- Established metrics to rate the interconnect
 - Latency
 - Bandwidth
 - Bisection bandwidth
 - Effective bandwidth
 - Effective bisection bandwidth

Algorithm 3 SSSP routing algorithm (Phase 2)

Input: Context of DFSSSP routing

Output: Linear Forwarding Tabellen

/ N-to-N, multi-graph Dijkstra algorithm */*

for all Port \in Subnet **do**

 Dijkstra(...) for this port as source

 Update all linear forwarding tables

 Increase edge wights

end for
