Layered Shortest Path (LASH) Routing in Irregular System Area Networks

Tor Skeie, Olav Lysne and Ingebjørg Theiss
Simula.Research Laboratory
P.O. Box 134, N-1325 Lysaker, Norway
{tskeie | olav.lysne | theiss}@simula.no

Abstract

In recent years we have seen a growing interest in irregular network topologies for cluster interconnects. One problem related to such topologies is that the combination of shortest path and deadlock free routing is difficult. As a result of this the existing solutions for routing in irregular networks either guarantee shortest paths relative to some constraint (like up/down*), or have to resort to deadlock recovery through non-minimal escape channels. In this paper we propose a method that guarantees shortest path routing and in-order delivery, and that uses virtual channels for deadlock avoidance. We present a theoretical upper bound on the number of virtual channels needed, and through extensive empirical testing we demonstrate that the actual number of virtual channels is very low even for large networks.*

1 Introduction

Since the seminal work of Kermani and Kleinrock on *virtual cut through* [11] and later Dally and Seitz on *wormhole routing* [4, 5] we have seen an ever increasing body of research on these routing techniques. They are now the predominant paradigms in multicomputer interconnection. For a survey of interconnection networks we refer to [6].

In recent years we have also seen that powerful workstations or PCs interconnected by off-the-shelf network switches based on these routing techniques have been introduced to the multiprocessor market. Such networks of workstations (NOWs) have the advantage that they offer greater flexibility than custom built multiprocessors, and frequently to a lower cost since they are built from off-the-shelf components. Examples of commercially available switches for use in NOWs are [2, 3, 7, 9, 10].

One property of NOWs is that their routing functionality must be able to handle irregular topologies as well as regular ones. A well-known method for generation of routing algorithms in irregular networks is the up*/down* routing, described in connection with [7]. It relies on a technique for generating a breadth-first spanning tree of the network, where the key issue is to determine directions of the links in order for the network to be deadlock free. Sancho and Robles propose an improvement of the up*/down* routing scheme in [15], where they apply a depth-first search that allow for greater flexibility when forming the underlying routing graph. In [13] Lysne and Skeie also discuss an improvement of the up*/down* algorithm by having multiple roots and in this way be able to spread the traffic, gaining significant increase in throughput (i.e. avoiding the hot spots around the single root of up*/down* routing). Quao and Ni takes a different approach to routing of irregular networks. Their algorithm is based on Eulerian trails [14]. Their method assumes an underlying Eulerian graph and can route adaptively between two acyclic unidirectional trails (paths) that contain all the channels (edges) of the network. In order to have more optimized paths, different types of short-cut channels are added following heuristic criteria.

Another body of work has focused on improving the performance of irregular networks through various methods like virtual channel multiplexing, adaptivity, and shortest path routing combined with escape paths. [17, 16, 18]. Potentially deadlocked packets (possibly determined by timers) are re-directed to the escape virtual layer, and must stay in this layer towards their destination in order to guarantee freedom from deadlocks.

One weakness of all the approaches is that they require extra functionality in the switches that not all technologies provide. In particular, methods based on adaptivity in the switches lead to out-of-order delivery of packets. The extra protocol overhead involved in sorting the packets at the destination is in some cases unacceptable. This is the reason why some technologies, like those based on the recent InfiniBand^{TM} specification [1], only use deterministic routing. Another weakness of the above approaches is that they do not guarantee shortest path routing of all packets. A source routed shortest path algorithm is studied in [8], where the temporary removal of packets at critical intermediate hosts avoids deadlock.

In this paper we present LASH routing, a method for deterministic shortest path routing of irregular networks in which all packets are routed minimally for any irregular topology, and in which all packets are delivered in-order. The concept assumes the presence of virtual channels divided into virtual networks (layers) to avoid deadlock. Other than that, LASH routing requires no special functionality within the switches, making it implementable within the InfiniBand$^{\mathrm{TM}}$ architecture. Achieving deadlock free, minimal, and adaptive routing using virtual channels was done for regular networks in [12], in particular for k-ary n-cubes. This method was impractical, because the required number of virtual channels per link grew exponentially with n. We show that in practice, the number of virtual channels in LASH routing is scalable with respect to network size.

The paper is organized as follows: Section 2 gives some basic definitions and notation, section 3 presents the algorithm for generating deadlock free minimal and deterministic routing functions using virtual layers. Section 4 is devoted to the development of a theoretical upper bound for the number of virtual channels needed. Simulation experiments to test the practical number of virtual layers the various topologies needed are described in section 5. Section 6 gives some hints on how to implement LASH in InfiniBand$^{\mathrm{TM}}$. Finally, we conclude in section 7.

2 Preliminaries

The definitions in this section adhere to the standard notation and definitions of cut-through switching and graph theory.

Definition 1 *An **interconnection network** I is represented by a strongly connected directed graph, $I = G(N, C)$. The vertices of I are the set of nodes (switches) N, whereas the edges are the set of communication channels (possibly virtual), C. Each channel is unidirectional and transmits data from a source node to a destination node. A network channel c_i interconnects two nodes $src(c_i)$ and $dst(c_i)$ $\in N$, the source and destination of the channel respectively. Two disjoint sets of channels C_k $(c_{k_1}, c_{k_2}, ..., c_{k_n})$ and C_l $(c_{l_1}, c_{l_2}, ..., c_{l_n})$ that interconnect two nodes, so that $src(c_{k_1}, c_{k_2}, ...) = dst(c_{l_1}, c_{l_2}, ...)$ and $src(c_{l_1}, c_{l_2}, ...) = dst(c_{k_1}, c_{k_2}, ...)$, is called a bi-directional link. A network is bi-directional if all its interconnections consist of such links.*

Definition 2 *A (deterministic) **routing function** $R : N \times C \times N \longrightarrow C$ takes a node n_i, an input channel c_{i_j} and a destination address n_d as parameters, and returns the output channel to be taken from node n_i for packets entering its channel c_{i_j} and whose destination is n_d.*

Notice that this definition of a routing function allows a node to select different output channels depending on what input channel a packet arrives on. This is not possible in technologies such as InfiniBand$^{\mathrm{TM}}$, but also is not necessary in the method we present in this paper. Our requirements are only that the routing tables are able to implement shortest path routing, and that the virtual layer on which each packet is to be routed can somehow be coded into the packet.

Definition 3 *For a network I and routing function R there exists a **dependency** from channel c_i to c_j iff $c_j = R(dst(c_i), c_i, n)$ for some node n. That is, packets destined for n may use c_j immediately after c_i.*

The following theorem is due to Dally and Seitz [5]. We will use this theorem in order to prove that the routing functions we define in this paper are deadlock free.

Theorem 1 *A wormhole network is **free from deadlocks** if its channel dependency graph is acyclic.*

3 LASH Routing

In this section we present the ideas and principles of our approach to LAyered SHortest path (LASH) routing. Freedom from deadlocks is attained by dividing the traffic into different virtual layers using virtual channels. The routing function R is defined by two sub-functions - R_{phys} and R_{virt}, respectively. The former defines one minimal physical path for each <source,destination> pair. The latter determines on which virtual layer (set of channels) packets from each <source,destination> pair should be forwarded along the minimal paths specified by R_{phys}.

The idea is that each virtual layer i in the network has a set of <source,destination> pairs R_{virt_i} assigned to it, in such a way that all <source,destination> pairs are assigned to exactly one virtual layer. In addition we make sure that each virtual layer is deadlock free by ensuring that the channel dependencies stemming from the <source,destination> pairs of one layer do not generate cycles.

The number of <source,destination> pairs in a network grows with the square of the number of nodes. We use the term *granular unit* (gu) to designate a set of <source,destination> pairs. Instead of adding <source,destination> pairs to virtual layers one by one, a gu is assigned en bloc.

Below we give an algorithm that assigns each <source,destination> pair to virtual layers.

Step 1: Obtain R_{phys} by finding the shortest path between any <source,destination> pairs within the network.

Step 2: Divide the set of all <source,destination> pairs into a set of *gus* in such a way that all *gus* by themselves are deadlock free.

Step 3: Take one gu of (source, destination> pairs that has not yet been assigned to a virtual layer. Find an existing virtual layer i such that gu can be added to R_{virt_i}, without closing a cycle of dependencies in virtual layer i. Add gu to R_{virt_i} (Basically, this step verifies that virtual layer i remains free from deadlocks).

Step 4: If step 3 is unsuccessful, create a new virtual layer j and let R_{virt_i} equal gu.

Step 5: If there are more gus that have not been assigned to a virtual layer, goto step 3.

Step 6: Obtain R_{virt} from the sets R_{virt_i} for all i, by letting all packets from each <source,destination> pair in R_{virt_i} be routed on virtual layer i.

Lemma 1 *The resulting routing function R is minimal and deadlock free.*

Proof: Since R_{phys} defines the physical paths of the packet forwarding, it follows that R is minimal. From steps 2 and 3 it follows that the channel dependency graph associated with each virtual layer i is acyclic. Since no packets are allowed to switch between virtual layers, the resulting channel graph is also acyclic. It follows from theorem 1 that R is deadlock free. □

From a computational point of view the main processing part of the algorithm is step 3. For each granular unit we examine if one or more channel dependency graphs are acyclic. The cost of searching for cycles in such a directed graph is bounded upwards linearly with the number of links in the network. For networks of reasonable size, the execution time is insignificant for the types of granular units we have used in our experiments (see section 5.1). Bigger topologies can be handled by increasing the size of the gus.

4 An Upper Bound on Virtual Layers

The most critical question related to the method is how many virtual layers are actually needed. This has implications to how the method scales relative to the size of the networks (number of nodes) and with respect to connectivity (number of links).

We start by discussing connectivity. When n switches are interconnected by $n-1$ or $n(n-1)/2$ bidirectional links one virtual layer will always suffice, if we assume that no link connects to the same pair of switches. The former characterizes a network with minimal connectivity, which means that it is structured as a tree. Since there are no loops in the network, there can be no cycles in the dependency graph with minimal routing. The latter case characterizes a network with maximal connectivity. Since all packets travel only one hop, this network will not have any channel dependencies at all. Between these two extremities the required number of virtual layers follows a curve that is first rising, and then falling. As we shall see in a later section, the peak appears to be around $2n$ for most network sizes.

Let us now turn our attention to the needed number of virtual layers as a function of the number of switches.

Lemma 2 *Assume a network of n switches interconnected by an arbitrary number of links, with a routing function R_{phys} that for each <source,destination> pair defines a minimal path through the network.*

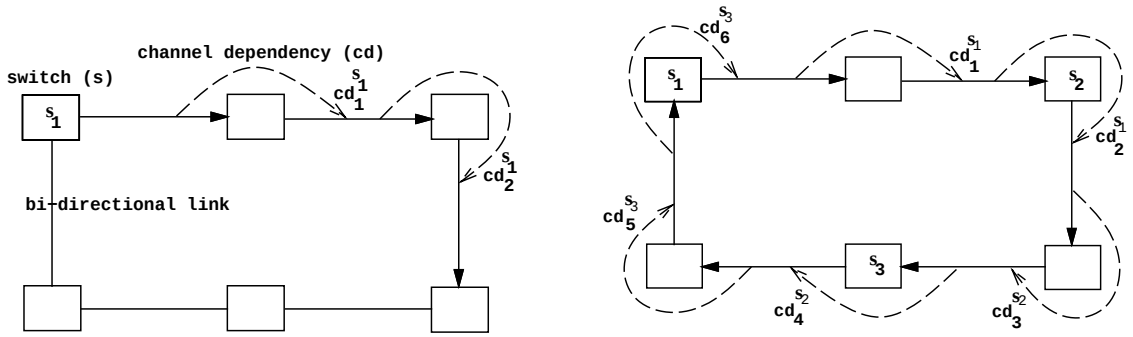
Then there exists an R_{virt} dividing <source,destination> pairs of the network into virtual layers using at most $\lceil n/2 \rceil$ layers, such that the routing function defined by R_{phys} and R_{virt} is free from deadlocks.

Proof: Let us divide the <source,destination> pairs in the network into gus such that each gu consists of all the shortest paths from one given node to all other nodes in the network. Obviously the number of gus will be n.

Consider an arbitrary gu consisting of all the shortest paths from a node N to any other node in the network. Assume that to each node N' in the network, we assign an integer $pathlength(N')$ corresponding to the length of the minimal path from N to that node. Now, for any two adjacent nodes, their pathlength-integers will either be identical, or one will be exactly *one* higher than the other (otherwise the integers could not correspond to shortest paths, as the one link between the two nodes could add at most one to the length of the minimal path).

Now consider any loop in the network, and any direction around that loop. For the gu to induce a dependency between two links going from $N1$ to $N2$ and from $N2$ to $N3$, the integers assigned to the three nodes must give $pathlength(N1) < pathlength(N2) < pathlength(N3)$. Since the loop ends up in $N1$ again, no more than half of the path around the loop can give us rising pathlength-integers, thus if there are I nodes in the loop, the number of dependencies in one direction around the loop provided by one gu will be less than $I/2$.

Since a cycle of dependencies around that loop will require I dependencies, and any one gu can contribute with less than $I/2$ dependencies, there will be room for at least two gus in each virtual layer. The fact that in our case there are n gus, gives that at most $\lceil n/2 \rceil$ virtual layers are needed. □



(a) Dependencies in a clockwise direction caused by a gu with switch s_1 as source.

(b) Dependencies in a clockwise direction caused by three gu with sources s_1, s_2, and s_3 respectively. As we can see, these dependencies form a cycle.

Figure 1. *A ring consisting of* 6 *switches and the channel dependencies created by the shortest paths from the switches* s_1, s_2, *and* s_3 *to other switches in the clockwise direction.*

However, under normal circumstances the needed number of virtual layers is very often far below our theoretical upper bound. For large networks the maximum number of virtual layers we have found necessary for any of the topologies we have generated is not even close to this bound.

The reason for this might be that we have not found the best possible (least) upper bound. Our bound is based on some of the properties of one algorithm for dividing traffic to layers. A search for the least upper bound, and for algorithms that generate routing tables that stay below this upper bound in terms of virtual layers, is a topic for further study.

One promising approach is to carefully choose the sequence in which the granular units are considered. In figure 1 we see an example network in which a cycle of dependencies may be closed already at three granular units. By choosing other gus than those represented by the sources s_1, s_2, and s_3 the cycle would not have been closed. In particular, if we considered gus with adjacent sources, the network in figure 1 would give room for four gus in one layer without closing a cycle of channel dependencies. It is easily shown that by considering gus with adjacent sources, a topology consisting of a single ring will in LASH routing only need two virtual layers.

5 Simulation experiments

To verify the practicality of the shortest path routing concept we have carried out an extensive set of experiments. The method has been tested for network sizes of 8, 16, 32, 64, and 128 switches. For each of these network sizes the link connectivity has been varied from $n-1$ (recall that this connectivity specifies a tree) and upwards, and for each of these network cases and connectivities, 100 random topologies have been generated.

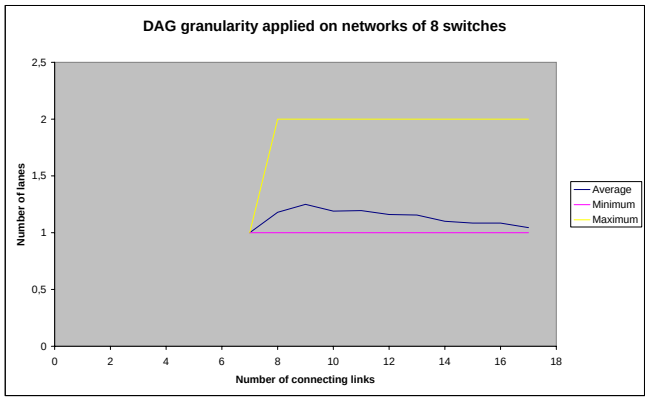
In our simulation experiments two classes of granular units in the allocation process of virtual channels were considered: i) *Directed Acyclic Graph (DAG) granularity* and ii) *Single Source-Destination (SSD) granularity*. The defined granular units gave different results with regard to the required number of virtual layers as network size increased. On the other hand, they also had different needs for execution time.

5.1 DAG granularity

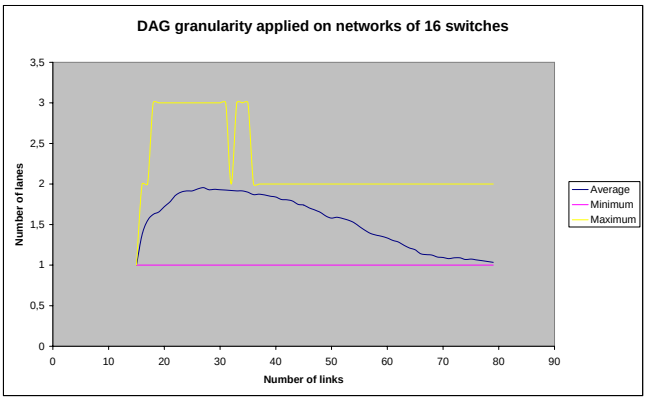
By DAG granularity we mean that one gu consists of all the shortest path from one given node to all other nodes in the network (this relates to the assumption in the proof of lemma 2). These paths define a DAG where the node under inspection is the root vertex of the graph (figure 2).

The results are displayed in figure 3. For all the tested network sizes the required number of virtual layers start off from 1 and increases sharply towards a peak, before they drop slowly towards 1 as the networks get increased connectivity. It appears that the maximum number of virtual layers are needed when the networks have twice as many links as switches.

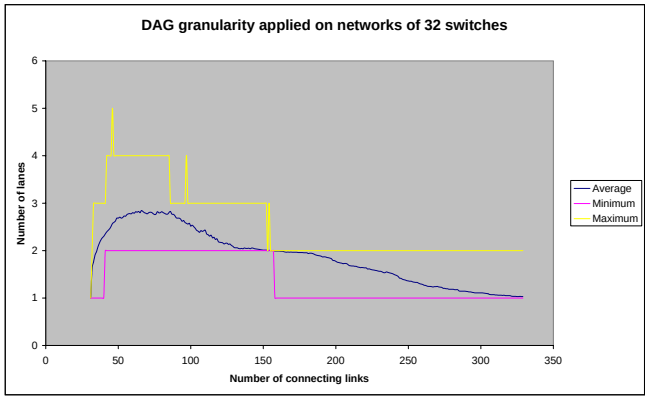
From the results we can also see that the method scales very well - in fact the ratio between network size (number of switches) and required number of virtual layers increases as the network size grows. For networks with 16, 32, 64, and



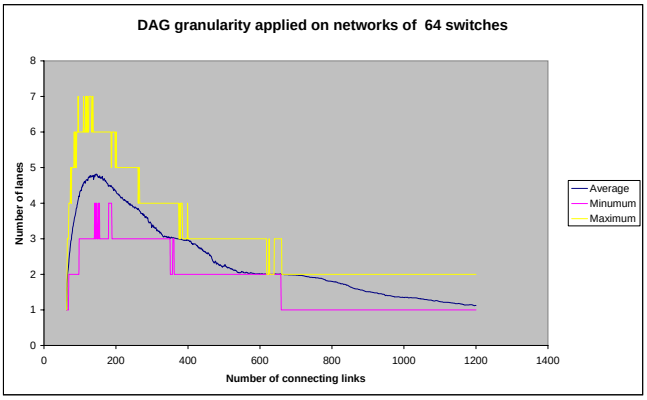
(a) Networks with 8 switches.



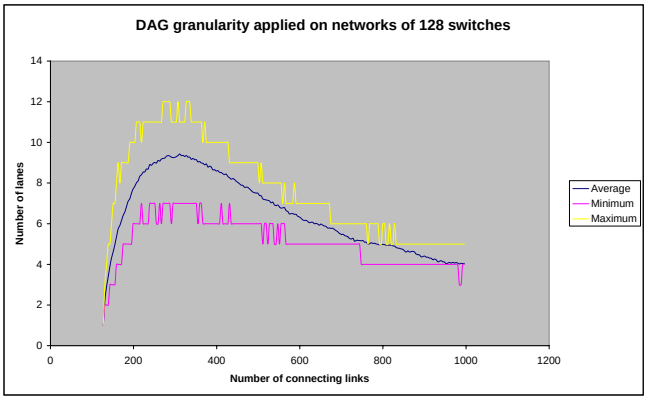
(b) Networks with 16 switches.



(c) Networks with 32 switches.



(d) Networks with 64 switches.



(e) Networks with 128 switches.

Figure 3. *These plots show the number of virtual layers needed for networks of various sizes when using DAG granularity of the gus. For each network size and link connectivity 100 random topologies were generated.*

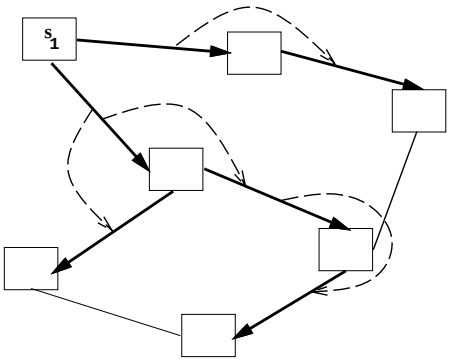


Figure 2. *An example of a DAG formed by the shortest paths (solid arrows) between a given switch and all other nodes. The associated dependencies of the DAG granular unit are shown as dotted curved arrows.*

128 switches the ratio figures are 8.4, 11.4, 13.3, and 13.6, respectively. Even for large networks with 128 switches the needed number of virtual layers, having an average peak of about 9, is acceptable - especially if we relate this number to the 15 virtual channels (for data traffic) offered by the InfiniBand™ standard. Notice also that we never experienced any randomly generated topology that needed more than 12 virtual layers. The upper bound given by our lemma is 64 layers for networks with 128 nodes, so this is a strong indication that our upper bound can be improved.

5.2 SSD granularity

In the SSD experiments, we let the granular units consist of only one single <source,destination> pair. This is in contrast to the DAG granularity where the gus hold multiple paths. The hypothesis of this experiment was that with a finer granularity our method should require fewer virtual layers, because generally there is a higher probability for creating cycles after adding the channel dependencies associated with an entire DAG than adding those provided by one single path. The results from the experiments, where we have considered networks consisting of 8, 16, 32, 64, and 128 switches, are shown in figure 4. Comparing these results with the ones from the DAG granularity (figure 3), we find that for small network sizes (consisting of 8 and 16 nodes) the methods perform equally well with respect to the needed number of virtual layers. However, as the network size grows, the difference becomes more noticeable. For networks of 32 and 64 switches the difference in average (maximum) need of virtual channels are about 1 (2) and 2 (4) at the most. But still, the figures are not very different. Note that for 64 nodes, connectivities above 250 were not tested for SSD.

An explanation for this similarity is that a DAG defines a tree structure. Subparts of the individual paths represented by the DAG typically represent a trunk of <source,destination> pairs that generate the same channel dependencies. Therefore, splitting the DAG into single <source,destination> pairs does not necessarily give substantial gain in the number of virtual layers needed. In terms of computing time of the routing algorithm, however, the difference is more significant for the following reasons:

- For SSD granularity $n \times (n-1)$ granular units must be processed by step 3 in the algorithm, in contrast to n units for the DAG option.
- The SSD approach builds up channel dependency graphs of higher connectivity than the DAG method. Since searching for cycles depends on the number of edges in the graph, this has impact on execution time.

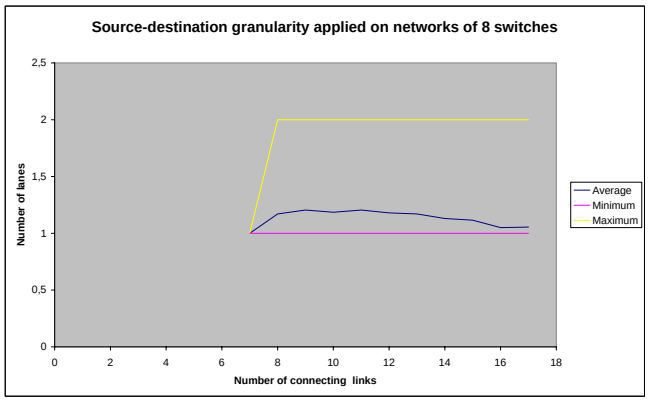
6 LASH in InfiniBand™

Above, we indicated that InfiniBand™ is a possible target technology for LASH routing. In InfiniBand™, switches only makes routing decisions based on the current switch and the destination. The partial minimal routing function R_{phys} can be implemented by choosing a singular outlink per destination on each switch. All packets going sharing destination, that also visits the same intermediate switch, will from this switch follow the same route. This is straightforward, but doesn't exploit multiple paths between two switches. Alternatively, using multiple ID's for each destination, the ability of a switch to choose output port based on inlink and/or source can be emulated. R_{virt} can be implemented by using the service level mechanism. The specification suggest using this mechanism for QoS levels, improved fabric utilization and deadlock avoidance [1]. Our method addresses the two latter of these, intending to leave as many service levels as possible for QoS.

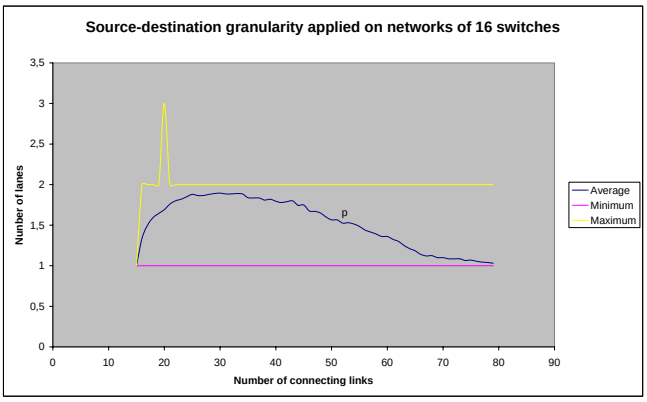
Up to 15 virtual lanes are available for the service levels, so our theoretical upper bound implicates that minimal deterministic routing with in-order delivery in any subnet with 30 or less switches can be achieved. In our experiments we never encountered cases that needed more than 12 virtual layers, which is very acceptable for InfiniBand™.

7 Conclusion

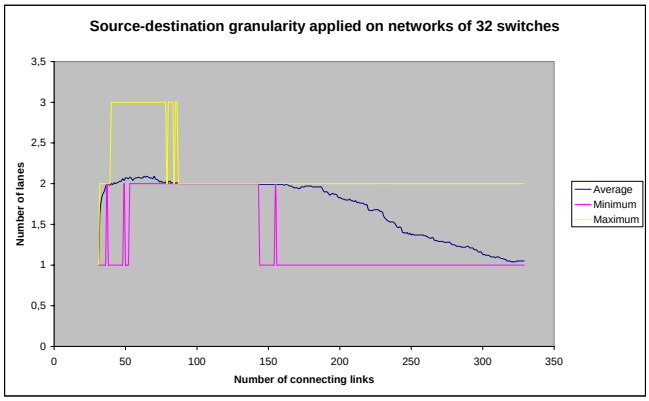
We have described a concept for shortest path routing of irregular networks that we call LASH routing. The concept relies on virtual channels, but otherwise requires no special functionality within the switches. In particular it is applicable to technologies like InfiniBand™ that cannot use adaptive routing together with escape paths.



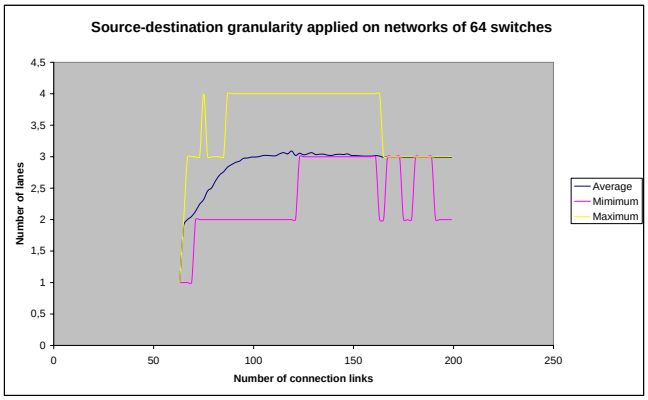
(a) Networks with 8 switches.



(b) Networks with 16 switches.



(c) Networks with 32 switches.



(d) Networks with 64 switches.

Figure 4. *The number of virtual layers needed for networks of various sizes under Single source-destination granularity. 100 random topologies were generated for each network size and link connectivity .*

Freedom from deadlock is achieved by dividing the physical network into a set of virtual layers. The minimal paths are spread onto these layers, such that each layer becomes deadlock free. A theoretical upper bound of needed number of virtual layers is given. Through extensive experiments generating random networks of different size and connectivity we see that the concept scales very well and that the required number of virtual layers is limited. For networks connected by 32, 64, and 128 switches applying the DAG granularity the average peak of needed virtual channels are 2.8, 4.8, and 9.1. The spread is small, and we have never needed more than 12 layers.

There are several directions for future research. First, we would like to find out how our method compares to state-of-art methods with respect to network performance. Another line of work is trying to optimize the process of determining the virtual routing function. We have seen that the method yields better results with granular units consisting of single <source,destination> pairs than with DAGs, but on the other hand, SSD has a higher computational complexity. One possible solution could be to make a hybrid of these two methods that produces nearly as good results as the SSD technique, but to the computation cost of the DAG. A somewhat different approach would be to do a network topology analysis, considering the physical loops in the network and study if the potential cycles here could be removed in a systematic way.

References

[1] I. T. Association. InfiniBand architecture specification, 2000.

[2] 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. IEEE MICRO, 1995.

[3] J. Carbonaro and F. Verhoorn. Cavallino: The teraflops router and NIC. In *Proceedings of Hot Interconnects IV*, pages 157–160, 1996.

[4] W. J. Dally and C. L. Seitz. The torus routing chip. *Distributed Computing*, 1:187–196, 1986.

[5] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, 1987.

[6] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks an engineering approach*. IEEE Computer Society, 1997.

[7] M. D. S. et.al. Autonet: a high-speed, self-configuring local area network using point-to-point links. SRC Research Report 59, Digital Equipment Corporation, 1990.

[8] J. Flich, M. P. Malumbres, P. Lòpes, and J. Duato. Performance evaluation of a new routing strategy for irregular networks with source routing. In *Proceedings of International Conference on Supercomputing*, May 2000.

[9] M. Galles. Spider: A high speed network interconnect. *IEEE Micro*, pages 34–39, Feb. 1997.

[10] R. W. Horst. Tnet: A reliable system area network. *IEEE Micro*, 15(1):37–45, 1995.

[11] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[12] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on Computers*, 40(1):2–12, 1991.

[13] O. Lysne and T. Skeie. Load balancing of irregular system area networks through multiple roots. In *Proceedings of 2nd International Conference on Communications in Computing*, 2001.

[14] W. Qiao and L. M. Ni. Adaptive routing in irregular networks using cut-through switches. In *Proceedings of the 1996 International Conference on Parallel Processing (ICPP '96)*, pages 52–60. IEEE Computer Society, 1996.

[15] J. C. Sancho, A. Robles, and J. Duato. A new methodology to compute deadlock-free routing tables for irregular networks. In *Proceedings of the Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing (CANPC'00)*, 2000.

[16] F. Silla and J. Duato. Improving the efficiency of adaptive routing in networks with irregular topology. In *Proceedings of the 1997 Int. Conference on High Performance computing*, 1997.

[17] F. Silla and J. Duato. On the use of virtual channels in networks of workstations with irregular topology. In *1997 Parallel Computing, Routing and Communication Workshop*, 1997.

[18] F. Silla, J. Duato, A. Sivasubramaniam, and C. R. Das. Virtual channel multiplexing in networks of workstations with irregular topology. In *Proceedings Fifth International Conference on High Performance Computing*, pages 147–154. IEEE Computer Society, 1998.