Boosting Ethernet Performance by Segment-Based Routing [*]

A. Mejia, J. Flich, J. Duato
Dept. de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
P.O.B. 22012, 46022 - Valencia, Spain
E-mail: {andres,jflich,jduato}@gap.upv.es

Sven-Arne Reinemo, Tor Skeie
Simula Research Laboratory
P.O.B. 134, N-1325
Lysaker, Norway
{svenar,tskeie}@simula.no

Abstract

Ethernet is turning out to be a cost-effective solution for building Cluster networks offering compatibility, simplicity, high bandwidth, scalability and a good performance-to-cost ratio. Nevertheless, Ethernet still makes inefficient use of network resources (links) and suffers from long failure recovery time due to the lack of a suitable routing algorithm. In this paper we embed an efficient routing algorithm into 802.3 Ethernet technology, making it possible to use off-the-shelf equipment to build high-performance and cost-effective Ethernet clusters, with an efficient use of link bandwidth and with fault tolerant capabilities. The algorithm, referred to as Segment-Based Routing (SR), is a deterministic routing algorithm that achieves high performance without the need for virtual channels (not available in Ethernet). Moreover, SR is topology agnostic, meaning it can be applied to any topology, and tolerates any combination of faults derived from the original topology when combined with static reconfiguration. Through simulations we verify an overall improvement in throughput by a factor of 1.2 to 10.0 when compared to the conventional Ethernet routing algorithm, the Spanning Tree Protocol (STP), and other topology agnostic routing algorithms such as Up/Down* and Tree-based Turn-prohibition, the last one being recently proposed for Ethernet.*

1 Introduction

Over the last years, clusters of PCs have become an attractive solution for building supercomputers. They provide an excellent performance-to-cost ratio when compared to proprietary solutions [28]. This trend has been facilitated by the emergence of high bandwidth and low latency network architectures such as Myrinet [5], InfiniBand [15], Quadrics [10], and, most recently, Advanced Switching Interconnect (ASI) [2].

Although conventional Ethernet can be used to build large systems, it has not been the preferred technology in high-performance computing (HPC) due to the following reasons: First, the high latency experienced by packets as a result of the different software layers a packet needs to cross (with the associated memory copies) has been deemed inappropriate for HPC. Second, the higher layer retransmission and acknowledgement protocols required by Ethernet networks increase latency. These protocols are required since Ethernet was designed as a lossy network where packets may be dropped in the presence of congestion, while advanced technologies like InfiniBand, Myrinet, and Quadrics are loss-less, which is more appropriate in HPC systems. Third, the lack of a routing algorithm suitable for HPC has compelled people to choose other alternatives. Fourth and finally, the lack of a congestion management mechanism in Ethernet has, in many cases, made it unattractive for HPC. This is, however, being addressed by the IEEE 802.3ar Congestion Management Task Force.

Fortunately, the situation is changing and the problems associated with using Ethernet for building HPC systems are slowly vanishing, and we can now find high performance commercial Ethernet switches in the marketplace. Zero copy protocols [3, 27] and remote memory access techniques [19] have been added to Ethernet interfaces in order to improve latency. Also, the addition of an on/off based flow-control mechanism to the Ethernet standard in 1997 [24], allows Ethernet to behave as a loss-less network [21] without the need for expensive retransmission and acknowledgement of packets. The only major problem left is the routing algorithm, which only supports the tree topology. If we can improve the routing algorithm, Ethernet could become more attractive for building HPC systems as it offers significant benefits. The most significant one is its low cost, which plays an important role for the size of a system. When compared to Myrinet and Quadrics, the cost of Ethernet is an order of magnitude lower [4, 8]. Furthermore, it is simple, scalable, widespread, and most existing applications are compatible with Ethernet, thus enabling a

[*] This work was supported by Junta de Comunidades de Castilla La Mancha under grant PBC-05-005-2. CONSOLIDER-INGENIO 2010 under grant CSD2006-00046 and CICYT under grant TIN2006-15516-C04-01

15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP'07)
0-7695-2784-1/07 $20.00 © 2007 IEEE

reduction in the development time for a new system.

1.1 Routing Algorithms for Ethernet

Routing in Ethernet networks is deterministic. That is, once a packet is injected, it follows a unique path until it reaches its destination. One of the main benefits of using deterministic routing is that in-order arrival of packets is preserved, which is usually required by HPC applications.

The standard routing algorithm for interconnecting conventional Ethernet is the *Spanning Tree Protocol* (STP) [14]. STP is a simple algorithm that works by turning any topology into a tree by disabling links (forcing redundant data paths into a standby blocked state) [25], therefore it can be viewed as a link prohibition mechanism. When designing a system for HPC it is desirable to use regular topologies such as meshes and tori in a loss-less, deadlock-free manner [1, 7, 9]. The STP, however, is inappropriate when using regular topologies, since the STP will end up building a tree out of any topology, thus disabling and wasting a large set of links throughout the network, which leads to low link utilisation and long failure recovery time.

In a flow controlled (loss-less) Ethernet, the routing algorithm must be carefully designed in order to avoid deadlock situations. A deadlock occurs when packets block the network indefinitely as they cyclically request resources used by other blocked packets (they form cyclic dependencies between channel resources [6, 7, 9]). The STP is not prone to deadlock as it converts any topology into a tree in order to avoid all loops in the network. But this leads to the drawback described above.

Several routing algorithms have been suggested for improving STP performance, such as SmartBridge [22], STAR [16], and OSR [12]. All of these strategies have been designed for lossy networks, without addressing the deadlock problem. Another, recent proposal, referred to as Viking [26], has a broader approach to the replacement of STP. In addition to targeting minimal routing, this scheme also provides load balancing of links and fault-tolerance in the case of a link failure. Viking exploits the IEEE 802.1Q Virtual Local Area Network (VLAN) technology [25] in combination with Multiple Spanning Trees by letting the VLAN tag decide which spanning tree to be used for routing. Although this method can be implemented without changes to the current standard, it requires a high number of VLANs and a Viking process running at each end-node. Moreover, as the standard does not allow for dedicated buffers per VLAN, the Viking algorithm may induce deadlock when flow control is used.

Another deterministic algorithm suitable for flow-controlled Ethernet technology is the *Up*/Down** algorithm (UD) [23]. UD can be used in any topology without the need for virtual channels (not supported by Ethernet), making it suitable for a wide range of network technologies, including Ethernet [21]. UD first selects a root node, then it computes a spanning tree, and finally it enforces *turn prohibitions* among the links to avoid cycles. A deterministic routing algorithm can be viewed as a set of turn prohibitions, where a turn is a ternary tuple consisting of an input link, a switch, and an output link [13]. A turn prohibition is a tuple that is forbidden by the routing algorithm. The routing algorithm restricts the number of turns allowed in the network in order to guarantee deadlock freedom and full connectivity. Therefore, UD allows the use of all links as it is based on turn prohibitions rather than link prohibitions as done by STP. UD, however, has another drawback as it is highly sensitive to hot-spots around the root of the spanning tree. Although this can be alleviated by the use of multiple roots as studied in [17], it requires the use of virtual channels which are not supported by Ethernet.

Some of the drawbacks of UD are addressed in [20], where an algorithm referred to as *Tree-based Turn-prohibition* (TBTP) is proposed. TBTP avoids the hot-spot problem and bounds the number of prohibited turns to a maximum of 50% independently of the topology used.

Even though many proposals exist, Ethernet still lacks an effective routing algorithm able to deliver maximum link utilisation without disabling links and able to balance traffic throughout the network. This is a challenging task since Ethernet networks usually are built with irregular topologies. In [18] we presented a deterministic, deadlock free, and topology agnostic algorithm called Segment-based Routing (SR) for loss-less interconnection networks. This algorithm uses a new approach for enforcing turn restrictions. In particular, SR places turn prohibitions in a distributed and independent manner thus allowing the routing algorithm to better exploit the underlying topology, with the benefit of increased performance and flexibility. In [18] the SR algorithm was designed to work with 2D mesh networks with failures.

In this paper we apply SR to flow-controlled Ethernet networks. As SR does not require the use of virtual channels, it could be though that SR is directly suitable to Ethernet. However, different issues must be addressed. To this end, in this work, we extend the SR algorithm in order to be used on Ethernet. In particular, we take on different challenges. First, we propose different ways of applying SR into current Ethernet switches with no hardware modifications. Secondly, we design computation rules in SR in order to be used on totally irregular networks. This is a challenging task since SR must obtain most of the available network bandwidth by achieving a good traffic balance from an irregular topology. Finally, we evaluate the routing algorithms that currently can be applied in Ethernet, that is, SR, TBTP and UD routings. Here we show the higher performance achieved by SR and its higher scalability in Ethernet networks.

The rest of this paper is organised as follows: Section 2 provides a detailed overview of the SR algorithm and how it can be used with Ethernet, while Section 3 presents simulation results for SR in comparison with UD and TBTP. In Section 4 two approaches to embed SR into Ethernet are discussed. And finally, in Section 5 some conclusions are given.

2 Segment-based Routing

The main goal of SR is to achieve high performance while providing connectivity among all the end-nodes and keeping the network deadlock free. Basically, the algorithm splits (if needed) the network in different subnets, and within each subnet it groups network components (switches and links) into disjoint segments. Thus, a subnet is formed by one or more segments and each network component belongs to a unique segment in a unique subnet (i.e. segments and subnets are disjoint). As an example, Figure 1(a) shows a topology plotting only switches (in circles) and links (in lines). From this topology, SR computes two different subnets with the segments described in Figure 1(b). In particular, 6 segments have been computed (from $S1$ to $S6$) [1]. A segment is defined as a list of interconnected switches and links. Segment $S1$ consists of switches $\{A, B, F, E\}$ and links $\{1, 2, 3, 4\}$. Segment $S2$ consists of switches $\{C, G\}$ and links $\{5, 6, 7\}$, while segment $S4$ is formed only by link $\{11\}$ and so on for the rest of the segments. Notice also that except the initial segment, the remaining segments start and end on a switch already part of a previously computed segment (e.g. $S2$ starts/ends from/on $S1$).

As soon as the complete topology is partitioned into segments, SR adds a bidirectional turn prohibition to every segment. By partitioning the network into independent segments, SR is able to place a turn prohibition in a segment independently of the remaining segments. Any possible combination will end up in a routing algorithm that is deadlock free and keeps connectivity among all end-nodes (see [18] for demonstration). Figure 1(b) shows all the possible turn prohibitions that can be placed on any segment. For example, in $S1$ we can place a bidirectional turn prohibition at switches B, E or F. From all the possible combinations, SR selects only one turn prohibition in each segment. As an example, we have placed one turn prohibition in each of the seven segments in Figure 1(c). Note that segment $S4$ received special treatment, as it is a unitary segment. In order to avoid cycles in the topology it is required that we block all traffic going across unitary segments. To achieve this, we select one out of the two switches attached to the unitary segment and place as many turn prohibitions as there are links connected to the already restricted segments. In the example we place two turn prohibitions at switch D.

Turn prohibitions can be placed at each segment independently from the selections made at the other segments. This property gives SR great flexibility when selecting the final set of turn prohibitions, which improves the final performance. For instance, in the example, there are 72 possible combinations ($3 \times 2 \times 2 \times 2 \times 3$) of turn prohibitions, whereas for UD routing there is only one possible combination after the root tree has been selected. This shows the flexibility provided by SR.

2.1 Detailed Description of SR

In this Section we provide a description of the SR algorithm. For a complete description with proofs, please refer to [18]. SR is based on three steps. First a segmentation of the topology is performed. Then, the selection of the turn prohibitions on each segment is done. Finally, SR selects the most suitable path between every source-destination pair.

In order to start with the segmentation process, SR labels each switch and link with a unique identifier. Then, it selects a switch to be the starting node. This switch is marked as visited, and from this node it searches for segments in the topology. The algorithm builds a segment by visiting switches and links until an already visited switch is found. Once the segment is found it is annotated, all its components are marked as visited, and from a visited switch a search for a new segment is started. This process is repeated until all links and switches have been visited.

In the process of searching segments, depending on the topology, it may happen that no new segment is found. For instance, in Figure 1(b), starting from switch H (already visited), the algorithm will search through link 17 ending at switch K not finding an already visited switch. In this case, we mark the switch from where we started the search as *Terminal* (T), and the link as *Bridge Link* (B), meaning that no new segments can be computed from them. This is an special segment where there is no need for a turn prohibition since it can not be part of a cycle. The same happens for switch J and link 15.

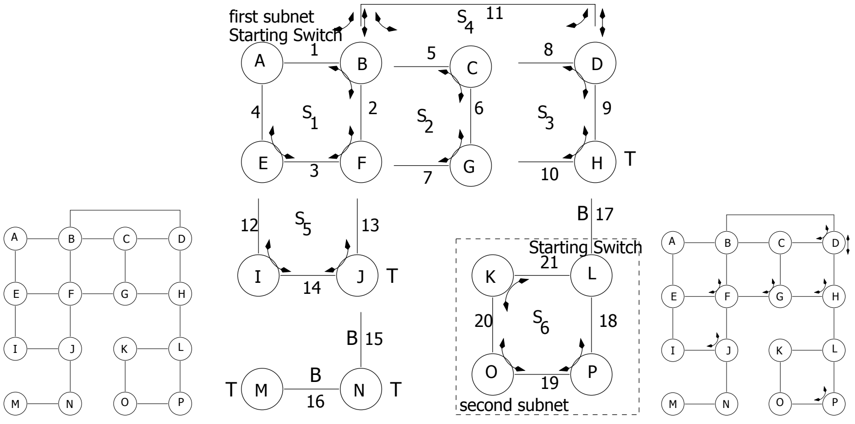
In addition to creating segments, the algorithm groups segments within subnets. A subnet is created when there is at least one switch connected to a bridge and the switch has one or more links that have not been visited. This switch becomes the starting switch in the new subnet, and we start looking for new segments until all links and switches have been visited.

When creating segments, each segment is classified into one of three types:

- **Starting segment**. This routing segment ($S1$ and $S6$ in the example) starts and ends on the same switch, thus forming a cycle. The starting segment is found every time a new subnet is initiated.
- **Regular segment**. This type of segment (e.g. $S2$ in the example) starts on a link, contains at least one switch, and ends on a link.
- **Unitary segment**. This type of routing segment (e.g. $S4$ in the example) contains only a link.

Once the routing segments have been computed and there are no more elements of the topology left to be visited,

[1] This topology has been selected in order to include all the special cases handled by SR.



(a) Topology (b) segments, subnets and possible prohibitions (c) Final

Figure 1. Example of computing SR algorithm.

the SR algorithm starts placing turn prohibitions on every segment. This is done in order to ensure deadlock freedom and at the same time to guarantee connectivity among all the end-nodes. As bidirectional links are used in the network, cycles must be broken on each direction, therefore we must place a bidirectional turn prohibition on each segment. SR will place prohibitions depending on the type of routing segment. In particular, for a starting segment, a bidirectional turn prohibition can be placed on any switch except the starting one (as it may introduce a cycle among different subnets). By doing this, we avoid all the cycles in the starting segment of the topology. For regular segments, one bidirectional turn prohibition can be placed on any of the switches belonging to the segment. This is done in order to break any possible loop crossing through the segment. Finally, for unitary segments, all the traffic crossing the link to an already restricted segment must be avoided in order to prevent deadlocks. Thus, in one side of the unitary segment bidirectional turn prohibitions must be placed for every link attached to the switch that connects to already restricted segments. Notice that unitary segments prevent most of the traffic to pass through the link.

Finally, in the third phase of the algorithm, paths for every source-destination pair are computed basd on the turn restrictions. The distribution of the selected paths will directly influence the performance obtained. The best combination of paths would be the one that offers minimal routing distance between every pair of nodes and balances the throughput efficiently throughout the network, but at the cost of increased computational cost. As stated in [18] the computational cost of the algorithm can be divided into three phases, in the first phase we use a smart calculation of segments with a cost of $O(m)$, where m is the number of links in the network. During the second phase, when turn restrictions are enforced the computational cost will be $O(s)$, where s is the total number of segments. Finally, in the third phase, paths are computed according to the turn restrictions. The complexity of this phase may vary greatly depending on the desired final performance. A straightforward random path selection will have a computational cost of $O(n^2)$ where n is the number of switches, while if an optimisation path selection algorithm is used the cost of this phase will be driven by the complexity of such an algorithm.

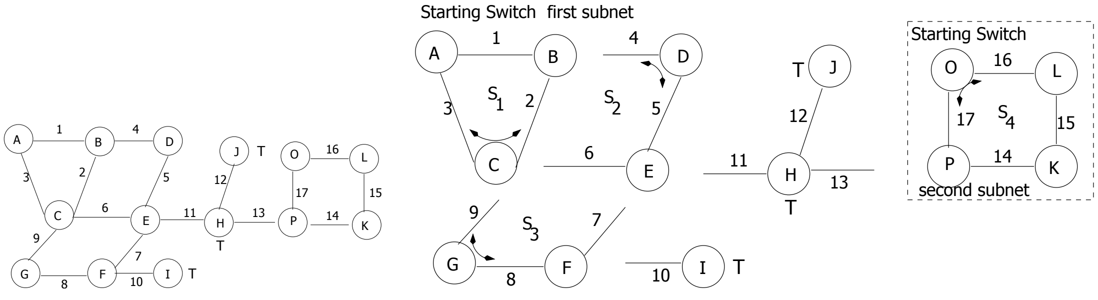
2.2 Applying SR on Irregular Networks

The selection of the initial switch and the order in which links and switches are searched influence the set of segments discovered, potentially having an impact on performance. For the case of 2D mesh networks with and without failures, a method of computing segments is provided in [18]. However, as Ethernet networks might use totally irregular topologies, we provide here a method for them.

The method will be described based on an example. Figure 2(a) shows an irregular topology. The switch with the lowest ID (A) is selected as the starting point for SR. From this switch, segments are computed by using the shortest distance to any already visited switch as a premise. Thus, segments are made as short as possible. By doing this, the number of unitary segments is reduced, which will lead to improved network throughput (as the number of turn prohibitions will be reduced.

SR computes two different subnets with the segments described in Figure 2(b). The first subnet contains three segments, segment $S1$ is the *starting segment* formed by switches $\{A, B, C\}$ and links $\{1, 2, 3\}$. Segment $S2$ is formed by nodes $\{D, E\}$ and links $\{4, 5, 6\}$ while segment $S3$ is formed by nodes $\{F, G\}$ and links $\{7, 8, 9\}$. Note that during the segmentation process segment $S2$ could have been formed by nodes $\{D, E, F, G\}$ and links $\{4, 5, 7, 8, 9\}$ but the algorithm chooses to go across link 6 instead of 7 when passing through switch E as it is looking for the shortest path to an already visited switch (C in this case). The second subnet contains only one segment, which is the starting segment ($S4$) of this subnet.

When placing prohibitions within a segment in an irregu-



(a) irregular topology

(b) segments, subnets and possible turn prohibitions

Figure 2. Example of computing SR algorithm in irregular networks.

lar topology, SR randomly selects the switch where to place the turn prohibition. Figure 2(b) shows how switch C has been elected to contain the turn restriction of $S1$, the same situation occurs to switches $\{D, G, O\}$ and its respective segments $\{S2, S3, S4\}$. Finally, source-destination paths have been calculated following the path balancing algorithm described in [11]. This method minimises the deviation of link weight.

2.3 Fault Tolerance Issues

SR provides fault tolerant features by providing more than one route between every *source-destination* pair, making it possible to immediately divert the affected routes in case of detecting failures on the network. SR precomputes (if possible) backup paths to every pair of *source-destination* switches. The backup paths need to be disjoint when compared to the main paths and provide the best possible balance of routes crossing a link. I.e. maximising the utilisation of low loaded links and minimising the use of heavily loaded links.

SR also needs an effective failure detection mechanism so that the switches can be informed about changes within a short duration of failure occurrence. For this purpose, SR relies on the failure detection support provided by Ethernet network switches. Each of the switches in the network is configured to send Simple Network Management Packets (SNMP) traps to the *manager switch* whenever any event of interest takes place. As the manager switch is responsible for the whole SR operation, we need to guarantee operation of this node. This can be achieved by having a backup switch ready to take over the role in the event of manager switch failure.

In the event of network failures (link failures, port failures, carrier loss, etc.), the failure detection mechanism running at each switch informs the manager switch, and the network enters a reconfiguration phase. The information is used by the manager switch to find out which paths are involved and what segment restrictions must be taken out in order to guarantee deadlock freedom and full connectivity of the network. When the decision is made, the involved switches are informed to activate backup paths, while new routing tables are recomputed and broadcasted to all switches in order to handle subsequent failures.

In order to provide immediate fault-tolerance, there should be at least two paths which do not share the faulty element of the network. One is the primary path and the other the backup path to switch when failure occurs. When there is not backup path for a *source-destination* pair the system is halted and drained of packets, then all routing tables are recomputed and distributed to all switches.

A similar operation takes place when adding a new segment to an existing network so SR retains the plug-and-play benefits supported by the STP. When a segment is added the manager switch is informed of the topology change and then recalculates and distributes routing tables to the new segment. Depending on the change, a new restriction might be needed in one or more of the already existing switches.

3 Performance Results

We present packet level simulations for a set of regular and irregular topologies. All simulation results have been obtained with an Ethernet simulator developed with the J-Sim framework [29]. We simulate a shared memory Ethernet switch with support for 802.3x flow control and 1 Gbit/s Ethernet links. Each switch has 5 ports where one is connected to a computing node and up to four are connected to other switches. Our traffic model consists of uniform and pairwise traffic patterns, and a peak rate packet arrival process. The average bit rate is increased in steps from 10 to 1000 Mbit/s (1% - 100% load). The packet size is fixed at 1522 bytes which is the maximum Ethernet frame size. We use Burton Normal Form [9] plots to visualise performance.

3.1 Regular Topologies

We have evaluated meshes and tori of sizes 4×4, 8×4 and 8×8. Due to lack of space we only show the average results for the 8x8 torus, the 8x8 mesh, and an 8x8 mesh with 5% random link faults. The latter is interesting as it shows the strength of SR when applied to semi-regular networks (regular topologies with link failures). We also simulate two different traffic patterns: Uniform and Pairwise distribution. For meshes and tori SR has been calculated following the procedure described in [18].

The results from an 8x8 torus in Fig. 3(a) show that SR outperforms all alternatives when uniform traffic is used. UD, being second best, is outperformed by a factor of 2.2. UD, TBTP and STP are all unable to effectively balance the traffic throughout the network as advantageously as SR does, which make them unable to reach the same level of performance. For the 8x8 mesh (Fig. 3(b)) the trend is maintained, now SR outperforms UD by a factor of 1.2. The situation has improved for UD since it takes advantage of the regularity of the topology, never the less it is affected by early saturation as a result of the hot-spots close to the root node. Finally, for a mesh with 5% link faults (Fig. 3(c)) SR outperforms UD by a factor of 2. Here, SR is able to retain much of the regularity of a full 8x8 mesh even in the presence of faults, while the alternatives perceive a significant reduction in performance. In general, STP and TBTP are unable to fully exploit the performance of the network since they do not distribute routes around the topology, but mainly use the links of the spanning tree which saturates quickly.

For pairwise traffic in Fig. 3(d), 3(e) and 3(f), we see similar results, but with an overall reduction in throughput as the pairs cause a non-uniform use of network resources causing an early saturation of the network.

3.2 Irregular Topologies

For irregular networks we have studied random topologies with 16, 32, and 64 switches. The evaluation of different sized networks gives us an indication of the scalability of each algorithm, as well as a performance evaluation on irregular networks. In Fig. 4(a) we see that SR increases throughput by a factor of 1.5 compared to UD, while UD and TBTP are similar in performance. For 32 switches (Fig. 4(b)) the performance difference is 1.84 in favour of SR compared to UD, and 2.4 in favour of SR compared to TBTP. So as the network size increase the performance of UD and TBTP is decreased compared to SR. For 64 switches (Fig. 4(c)) SR outperforms UD and TBTP with a factor of 2.0 and 3.2 respectively. SR both perform and scale better than the other alternatives. This is due to the advantage of having an even distribution of traffic across the network generated by its flexibility and its locality independence when placing turn prohibitions. This combination of a local (within a segment) and global (between segments) view of the network ensures better decisions when enforcing turn prohibitions compared to the global only perspective of UD and TBTP. And, as we saw above, this aspect becomes more important as the network size increase.

Again, the results are similar for pairwise traffic (Figs. 4(d), 4(e), and 4(f)), but with smaller differences due to the non-uniform use of the network.

Finally, let us point out that the use of turn-prohibition algorithms (SR, UD, TBTP) improves the performance of the network when compared with link-prohibition algorithms (STP), an effect due to the fact that we do not disable links.

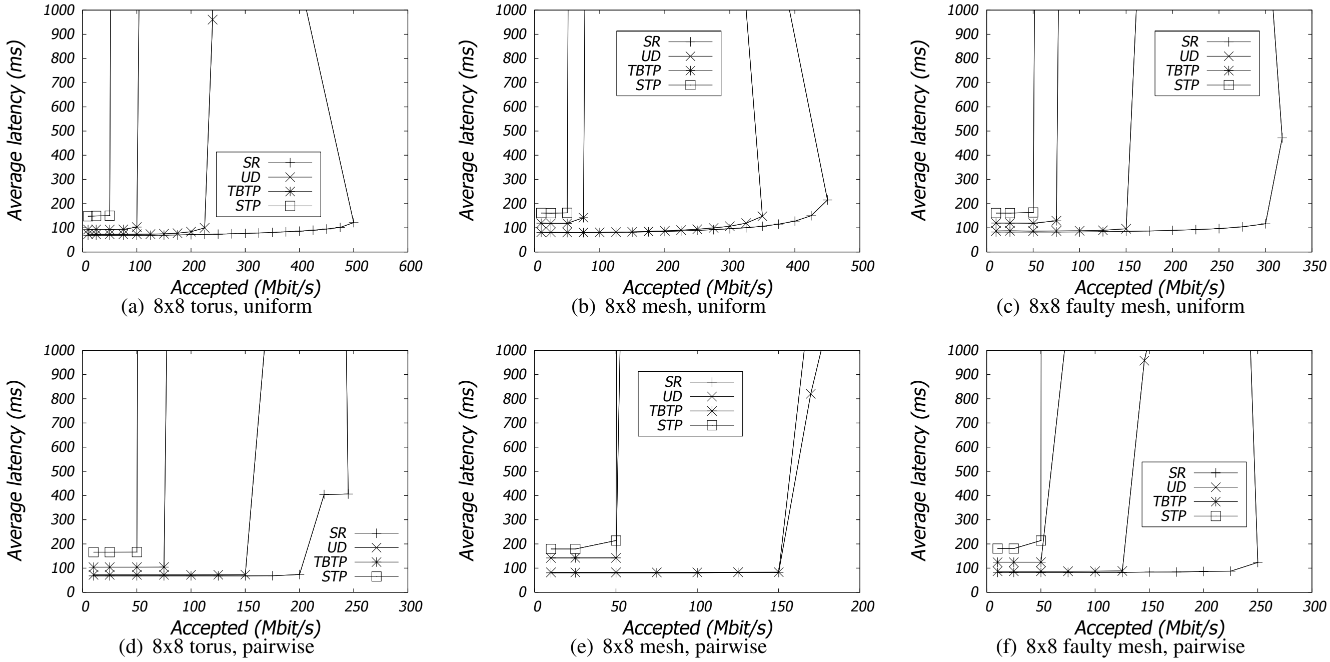
4 Embedding SR into Ethernet

The Spanning Tree Protocol is embedded in all conventional Ethernet switches and it makes the configuration of Ethernet equipment an effortless task. Every switch has an instance of the algorithm running and ready to perform the following operations: (a) Elect a *root switch*. The switch with the lowest identifier is selected as the root, and it becomes the root of the spanning tree. (b) Negotiate the deactivation of ports in order to build a spanning tree. This process avoids loops by making sure that only one switch is responsible for forwarding frames from the direction of the root on to a given link. (c) Listen for changes in the topology (maintenance mode). While in maintenance mode all switches are listening for topology changes, and when a change is detected a new round of negotiations begins. During the negotiation phase routing might be inconsistent.

In order to replace STP with SR we will consider two approaches. The first approach requires that the standard organisations embrace SR and that the firmware in future Ethernet switches are shipped with SR embedded. The second approach provides a less elegant solution, but with some effort it can be used in current off-the-shelf equipment.

Clearly, the best way to embed SR into Ethernet is to replace STP by SR, and deliver auto-configuration support at the same levels as STP currently does. This requires the presence of the SR algorithm in all switches and the support for the following operations: (i) Elect a master switch. (ii) Collect topology information. (iii) Calculate SR tables. (iv) Distribute routing tables. In step (i) the master switch is selected through negotiation as with the root switch in step (a) above. After the master switch is selected, it collects topology information (ii) from all other switches and creates a complete image of the network. Then, it calculates (iii) and distributes (iv) routing tables to all switches. Whenever the topology changes, step (ii), (iii), and (iv) have to be repeated. But as a result of SR's use of segments the number of switches involved in reconfiguration will be reduced because many changes can be handled locally within a segment. The downside of this approach is that we must convince the standard organisations to adopt SR.

Our second approach does not require any changes to conventional Ethernet switches and relies on features available in all *managed* Ethernet switches. A routing table for SR requires that the route look-up function is able to consider both the destination address and the input port when selecting the output port for the frame to be forwarded. A requirement that is met by most Ethernet switches classified as *managed* switches. This makes it possible to precalculate and then distribute routing tables to the switches involved by the use of the simple network management protocol (SNMP). The necessary tools will have to be written and the switches will have to operate in manual mode (i.e.



(a) 8x8 torus, uniform

(b) 8x8 mesh, uniform

(c) 8x8 faulty mesh, uniform

(d) 8x8 torus, pairwise

(e) 8x8 mesh, pairwise

(f) 8x8 faulty mesh, pairwise

Figure 3. Performance for regular topologies. Uniform and pairwise traffic.

STP and auto-learning have to be disabled). And, if the topology changes new routing tables will have to be uploaded. Clearly, this approach is somewhat cumbersome as we have no auto-configuration, but the performance gains are significant and can make it worthwhile for HPC systems.

5 Conclusion

We have proposed Segment-based Routing as a topology agnostic routing algorithm for Ethernet. Segment-based routing does not require virtual channels so it is easily implemented in Ethernet. Its novelty lies in the introduction of the *locality independence* property, which makes it possible to exploit the regularity of regular and semi-regular networks while at the same time achieving high performance for irregular networks. Furthermore, it improves scalability as the network size grows by combining local (per segment) and global (between segments) placement of turn prohibitions. This gives us a large degree of freedom when placing turn prohibitions and increase performance by efficiently using all links in the network.

References

[1] N. Adiga, M. Blumrich, D. Chen, and et al. Blue gene/l torus interconnection network. *IBM Journal of Research and Development*, 49, march 2005.

[2] ASI SIG. *Advanced Switching Core Architecture*. http://www.asi-sig.org.

[3] P. Balaji, S. Bhagvat, H. Jin, and D. Panda. Asynchronous zero-copy communication for synchronous sockets in the sockets direct protocol (sdp) over infiniband. In *In the proceedings of the 2006 workshop on Communication Architecture for Clusters (CAC); in conjunction with the IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006), Rhodes Island, Greece*, april 2006.

[4] B. M. Bode, J. J. Hill, and T. R. Benjegerdes. Cluster interconnect overview. In *Proceedings of the 2004 USENIX Annual Technical Conference*, pages 211–218. USENIX Association, july 2004.

[5] 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 lan. *IEEE MICRO*, 1995.

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

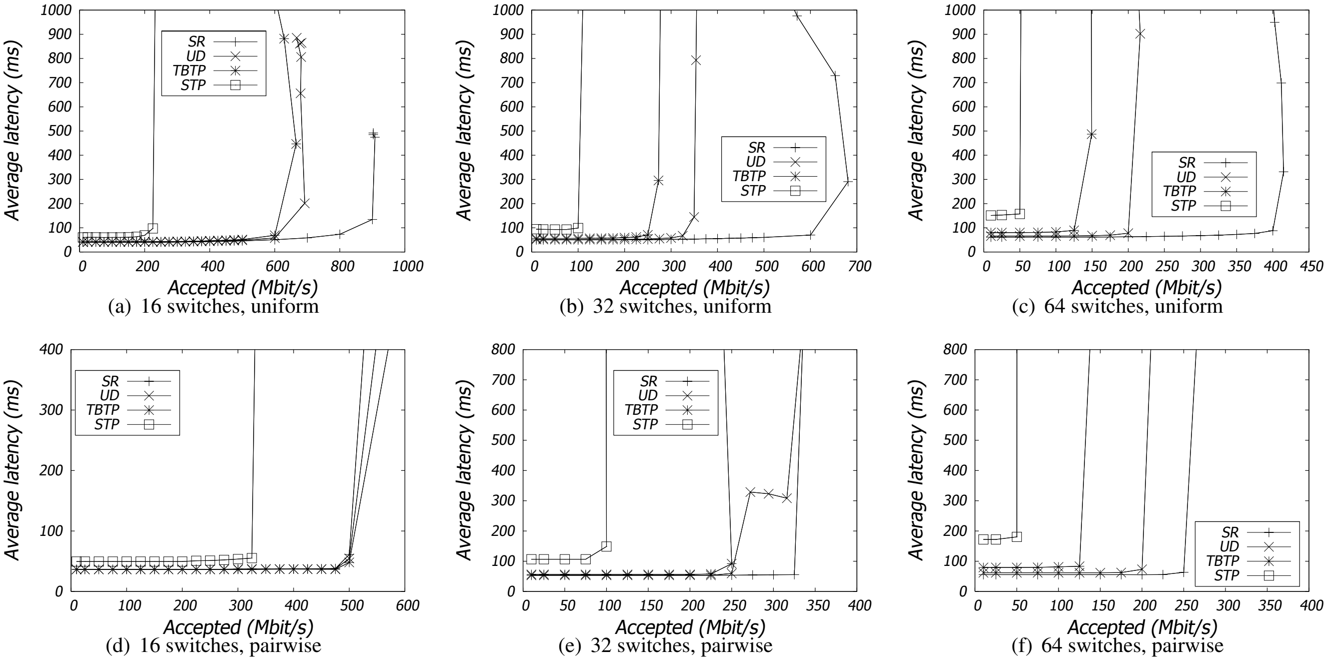
[7] W. J. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufman, 2004.

[8] R. W. Dobinson, M. Dobson, S. Haas, B. Martin, M. LeVine, and F. Saka. Ieee 802.3 ethernet, current status and future prospects at the lhc. In *Nuclear Science Symposium Conference Record, 2000 IEEE*, volume 3, 2000.

[9] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks an Engineering Approach*. IEEE Computer Society, 2003.

[10] F. P. W. Feng and A. Hoisie. The quadrics network (qsnet): high performance clustering technology. In *Hot Interconnects*, pages 125–130, Stanford, California, 2001.

[11] J. Flich, J. Duato, and et al. Combining in-transit buffers with optimized routing schemes to boost the performance of networks with source routing. In *2000 International Symposium on High Performance Computing*, Oct. 2000.



(a) 16 switches, uniform (b) 32 switches, uniform (c) 64 switches, uniform

(d) 16 switches, pairwise (e) 32 switches, pairwise (f) 64 switches, pairwise

Figure 4. Performance for irregular topologies. Uniform and pairwise traffic

[12] R. Garcia, J. Duato, and J. Serrano. A new transparent bridge protocol for lan internetworking using topologies with active loops. In *ICPP '98: International Conference on Parallel Processing*, pages 295–303, Washington, DC, USA, 1998.

[13] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *J. ACM*, 41(5):874–902, 1994.

[14] IEEE Standards Association. *ANSI/IEEE Std 802.1D Media access control (MAC) bridges*, 1998.

[15] InfiniBand Trade Association. *Infiniband architecture specification*, 1.2 edition, october 2004.

[16] K.-S. Lui, W. C. Lee, and K. Nahrstedt. Star: a transparent spanning tree bridge protocol with alternate routing. *SIGCOMM Comput. Commun. Rev.*, 32(3):33–46, 2002.

[17] O. Lysne and T. Skeie. Load balancing of irregular system area networks through multiple roots. In *Proceedings of the International Conference on Communication in Computing*, pages 165–171, june 2001.

[18] 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 *Proccedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)*. IEEE, april 2006.

[19] V. T. G. S. J. Nieplocha and D. Panda. Host-assisted zero-copy remote memory access communication on infiniband. In *International Parallel and Distributed Computing Symposium, IPDPS04*, Santa Fe, NM., 2004.

[20] F. D. Pellegrini, D. Starobinski, M. G. Karpovsky, and L. B. Levitin. Scalable cycle-breaking algorithms for gigabit ethernet backbones. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2175–2184, march 2004.

[21] S.-A. Reinemo and T. Skeie. Ethernet as a lossless deadlock free system area network. In *Parallel and Distributed Processing and Applications: ISPA 2005, Nanjing, China*, volume 3758, pages 901–914. Springer Berlin/Heidelberg, october 2005.

[22] T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson. Smartbridge: a scalable bridge architecture. *SIGCOMM Comput. Commun. Rev.*, 30(4):205–216, 2000.

[23] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, and T. L. Rodeheffer. Autonet: a high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications*, 9(8), october 1991.

[24] R. Seifert. *Gigabit Ethernet*. Addison Wesley, 1998.

[25] R. Seifert. *The Switch Book: The Complete Guide to LAN Switching Technology*. John Wiley & Sons, Inc., 2000.

[26] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. Viking: a multi-spanning-tree ethernet architecture for metropolitan area and cluster networks. In *INFOCOM 2004. 23th Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2283–2294, march 2004.

[27] P. Shivam, P. Wyckoff, and D. Panda. (emp): Zero-copy (os)-bypass (nic)-driven (gigabit ethernet) message passing. In *ACM/IEEE Conference on High Performance Networking and Computing*, pages 57–57, Denver, Colorado, 2001.

[28] top500.org. *Lists Top500 upercomputing Sites*, June 2006. www.top500.org.

[29] H.-Y. Tyan. *Design, Realization, and Evaluation of Component-Based Compsitional Software Architecture for Network Simulation*. PhD thesis, Ohio State University, 2002.