This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Effective Shortest Path Routing for Gigabit Ethernet

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

***Abstract*—Ethernet's native routing algorithm, the Spanning Tree Protocol, causes a major performance bottleneck when network connectivity increases. Since the Spanning Tree Protocol avoids deadlocks and infinitely looping packets by turning any topology into a tree, it leaves a large portion of links unused and thus wastes bandwidth. In this paper we address this weakness by proposing a new routing algorithm that avoids disabling links and prohibiting turns, and that guarantees shortest path routing.**

Through the use of *layered routing* we show how to improve performance with respect to both the Spanning Tree Protocol and a more recent proposal called Tree-Based Turn-Prohibition. The concept of layered routing is used to group virtual channels into network layers, and assign a limited set of source/destination address pairs to each layer. Extensive simulations show that we are able to increase throughput by a factor of more than 3.5 compared to the Spanning Tree Protocol and a factor of 1.8 compared to Tree-Based Turn-Prohibition. Our concept relies on features introduced in IEEE standards 802.1Q, 802.1D and 802.3x, as well as changes currently discussed in IEEE task forces.

I. Introduction

Ethernet is the dominating technology in local area and wireless networks, and with the recent introduction of 10 Gigabit Ethernet it is attractive in both system and wide area networking. Furthermore, the recent effort to standardise Backplane Ethernet will allow Ethernet to be used in server and I/O backplanes in the future.

There are, however, some challenges left before Ethernet can compete with high performance technologies such as Advanced Switching Interconnect [1], InfiniBand [2] and Myrinet [3]. These technologies all use credit-based flow control to avoid frame loss, and implement virtual channels (layers) for traffic isolation, advanced routing and QoS support. Flow control is mandatory to avoid frame loss, and especially important in a Storage Area Network where protocols such as iSCSI[4] and ATA over Ethernet[5] are deployed. Flow control also improves the performance of higher layer protocols such as TCP [6], [7], [8]. Virtual channels have many purposes and can be used for traffic segregation, QoS, and deadlock free shortest path routing [9]. Ethernet currently supports flow control and frame priorities , but lacks explicit support for virtual channels [10]. Lack of virtual channels manifests itself in many ways, the aspect of interest to us is efficient routing.

When designing a routing algorithm one of the base criteria is that it must possess deadlock freedom. The combination of topologies embedding loops and lossless flow control may lead to deadlocks. This phenomenon occurs when a set of packets are all stalled because all paths toward the destinations are blocked by another packet in the set, forming cyclic dependencies between channel (buffer) resources. Therefore, the routing strategies must be chosen carefully in order to avoid deadlock [11], [9].

Ethernet avoids deadlock through the use of the spanning tree protocol (STP) [10]. The spanning tree protocol reduce any topology to a tree by disabling links until we have no cycles left. This removes the deadlock potential, but it also removes a lot of bandwidth. Efficient topologies such as k-ary n-mesh and k-ary n-cubes contain lots of loops, and if we remove links the available bandwidth decrease.

In this paper we challenge Ethernet's poor routing algorithm and propose a deterministic, deadlock free, shortest path routing algorithm called LASH, which yields a performance increase by a factor of 3.8 when compared to the Spanning Tree Protocol, and a factor of 1.6 when compared to the more recent Tree-Based Turn-Prohibition algorithm [12].

In section II we give an overview of related work. In section III we describe the LASH routing algorithm. Its applicability to Ethernet is discussed in section IV, followed by a performance evaluation in section V. We conclude in section VI.

II. Related Work

Several algorithms which use some form of minimal routing have been suggested for the replacement of STP, such as SmartBridge [13], STAR [14], and OSR [15]. Common for all these strategies are that they do not address the deadlock problem relative to flow control. Furthermore, SmartBridge is not backwards compatible with existing Ethernet equipment, STAR does not guarantee shortest paths, and OSR requires changes to the Ethernet frame format or the use of tunnelling.

A more recent approach called Viking [16] 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 link failure. It achieves all this within the limits of the current standard, but it requires additional processes running at each end-node and a high number of VLANs (Virtual Local Area Networks [10]). Furthermore, as the VLAN principle does not support a dedicated buffer per VLAN, the union of all VLANs may deadlock if flow control is used.

Up/Down** is one of the better known algorithms that can replace STP. It was first described by Schroeder et. al in [17] and can be used with any topology without the need for virtual channels. This makes it suitable for a wide range of network technologies, including Ethernet [8]. It is a spanning tree based routing algorithm that allows the use of all links only constrained by turn prohibition [18], opposed to to the link prohibition used in STP. Up*/Down* has a major drawback in that it is vulnerable to hot spots around the root of the spanning tree, and since it is a tree based algorithm it does not necessarily allow for shortest path routing. Another drawback is the lack of an upper bound on the number of turns that are prohibited, which is dependent on the topology in question.

Some of the drawbacks of Up*/Down* are rectified by Pellegrini et al. in [12]. They propose an algorithm called *Tree-based Turn-prohibition* (TBTP), which avoids the hot spot problem seen in Up*/Down* and on average it prohibits 10% fewer turns. It also guarantees that a maximum of 50% of the turns in a topology are prohibited, independent of the topology in question.

The replacement of STP is currently being studied in the IEEE 802.1aq Shortest Path Bridging group, where they currently have suggested a solution based on multiple spanning trees. In this scheme each switch has its own spanning tree making shortest path routing possible, but it does not consider the deadlock issue, which gives it the the same weakness as Viking.

LASH avoids all of the above deficiencies by making deterministic, deadlock free, shortest path routing possible with only a very limited number of virtual channels. E.g. a network of 32 switches would on average require only two layers.

III. Layered Shortest Path Routing

Layered shortest path routing (LASH) is a minimal deterministic routing algorithm that guarantees shortest path routing and in-order delivery for both regular and irregular topologies [19]. The idea is that each virtual layer in the network has a set of source/destination pairs $\langle s, d\rangle$ assigned to it, in such a way that all $\langle s, d\rangle$ pairs are assigned to exactly one virtual layer. In addition it makes sure that each virtual layer is deadlock free by ensuring that the channel dependencies stemming from the $\langle s, d\rangle$ pairs of one layer do not generate cycles.

A. Layered Routing

Below we give a set of definitions for layered routing that adhere to the established definitions and notation of cut-through switching and graph theory.

Definition 1: A *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 unidirectional communication channels (possibly virtual), C. A network channel c_i interconnects the two nodes $src(c_i)$, $dst(c_i) \in N$, which are the source and destination of the channel, respectively. A *link* is a set of channels $c_{l_1}, c_{l_2}, ..., c_{l_n}$ such that either $src(c_{l_i}) = src(c_{l_j})$ and $dst(c_{l_i}) = dst(c_{l_j})$, or $src(c_{l_i}) = dst(c_{l_j})$ and $dst(c_{l_i}) = src(c_{l_j})$ for all i and j. Each channel $c \in C$ is part of exactly one link. A subset of the nodes N in the network are called *compute nodes*, these nodes generate and consume data traffic.

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.

Definition 3: The *channel dependency graph* of a network I with respect to a routing function R is a directed graph in which the channels of I constitute the vertices, and the dependencies constitute the arcs.

The following theorem is a straightforward adaptation of a theorem due to Dally and Seitz [11].

Theorem 1: A network is *free from deadlocks* if the channel dependency graph of its routing function is acyclic.

Definition 4: A *network layer* L_i of network I is a subset of the virtual channels in I such that each link has exactly two channels in L_i, one in each direction.

Definition 5: A set L of network layers $\{L_1, \ldots, L_n\}$ is a *layering* of a network I iff for $1 \leq i \leq n$ and $1 \leq j \leq n$

- L_i is a layer of I for all i,
- L_i and L_j are disjoint for all distinct i and j,
- for each channel c in I there exists an L_i such that c is in L_i.

The above two definitions allow us to view any layer of a network as a bidirectional virtual network that is isomorphic to the original physical network.

Definition 6: For a layering L of network I, and a routing function R, we say that R is *layered* with respect to L if $R(n, c_i, a) = c_j$ implies that c_i and c_j are in the same layer $L_j \in L$ for all n, c_i, a and c_j. By R_k we denote the subrouting function of R that is restricted to $L_k \in L$.
This means that a layered routing function will keep packets in the layer that it was first injected into. Furthermore, R_i contains all necessary information on the forwarding that can take place in L_i.

B. LASH Algorithm

The following algorithm can now be used to map $\langle s, d\rangle$ pairs onto virtual layers. We assume that a network I, and a layering L of that network is given. Furthermore, we assume that L has n layers. Let $T(\langle s, d\rangle)$ be the set of all layers that can be used for transmission between s and d.

Step 1: Let $T(\langle s, d\rangle) =$ *undefined* for all $\langle s, d\rangle$ pairs and let R_i be empty for all i such that $1 \leq i \leq n$.
Step 2: Take a source and destination $\langle s, d\rangle$ pair that has not yet been processed. For an arbitrary shortest path between this pair find an existing layer L_i such that letting R_i be enriched to support the path, and letting $T(\langle s, d\rangle) = \{L_i\}$ will not close a cycle of dependencies in the layered dependency graph of I. If one exists, let $T(\langle s, d\rangle) = \{L_i\}$, otherwise leave $T(\langle s, d\rangle)$ unchanged.
Step 3: If there are more $\langle s, d\rangle$ pairs that have not yet been processed, go to Step 2.

Lemma 1: If the above algorithm results in a $T(\langle s, d\rangle)$ which $T(\langle s, d\rangle) \neq$ *undefined* for all $\langle s, d\rangle$ pairs, the layered network I is free from deadlock with respect to R and T. Furthermore all packets are routed along shortest paths.

Proof: Shortest-path routing follows immediately from Step 2.

Assume that the lemma is not true. Then, the algorithm must in some case terminate with a deadlocked system. According to theorem 1 a cycle in the layered dependency graph must either have been there from the start of the algorithm or been introduced at some point. From Step 1 we can deduce that the layered dependency graph contains no dependencies, and therefore no cycles, when the algorithm starts, and hence that the cycle must have been introduced at a later stage.

The layered dependency graph is only altered by Step 2. The cycle in the layered dependency graph must, therefore, have appeared through the assignment of a layer to a $\langle s, d\rangle$ pair. By inspection of Step 2 we see that no such assignment will take place if it closes a cycle in the layered dependency graph. Thus, we have a contradiction, and the lemma follows. ■

The assignment of a path to a layer in Lemma 1 above is done by a random selection, this might not be optimum and could lead to insufficient traffic balancing. The advantage is that it is straightforward to implement, but it could be replaced by advanced schemes [20].

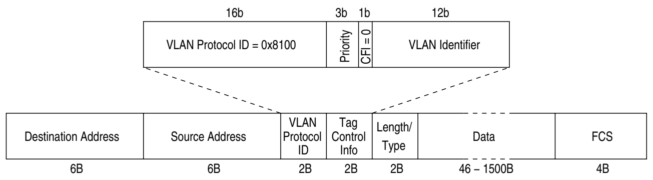
1) Required number of layers: An important issue in the evaluation of layered routing is the number of layers that are needed to grant shortest-path routing to every $\langle s, d\rangle$ pair. The required number of layers depends on both network size and connectivity. Note that the number of layers depends on the number of switches in the network, not the number of compute nodes (hosts). The number of compute nodes is only limited by the number of ports on each switch. For networks with 16, 32 and 64 switches and any connectivity, covering all $\langle s, d\rangle$ pairs (i.e. all switches) requires a maximum of three, three, and five layers, respectively. The average number of layers required are two, two and three, respectively. Studies show that that the number of required layers appears to follow a logarithmic curve [19].

Another surprising outcome is that the variance in the required number of layers is very small. For a set of 100 random topologies the difference between the most demanding and the least demanding topology was never more than one layer.

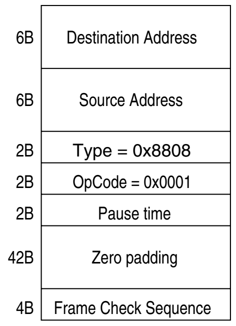
2) Computational Complexity: The complexity of the algorithm is given by the number of $\langle s, d\rangle$ pairs $|N|^2$ times the number of layers n, times the complexity of checking for cycles $|N|$, i.e. $O(|N|^3)$. This should not pose a problem for modern hardware when working with reasonable sized networks (< 256), and for larger networks optimisations might be possible.

IV. Ethernet Compatibility

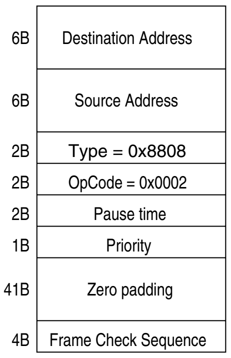
Below we briefly present the the priority and flow control mechanisms supported in Ethernet, and the changes necessary to implement the LASH algorithm.



(a) Data frame.



(b) Pause frame.



(c) New pause frame.

Fig. 1. Ethernet frame formats.

A. IEEE 802.1Q as Virtual Channels

The IEEE 802.1Q standard introduces priority tagging of Ethernet frames, where each frame can have a priority from 0-7, and each priority has dedicated buffering resources. We will exploit this feature as virtual channels. Figure 1(a) shows the format of a IEEE 802.1Q compliant Ethernet frame. The 3 bits labelled priority are used to indicate the priority of a given frame. To enable the concept of virtual channels we change the semantics of these three bits from priority to *virtual channel identifier*, which allows for eight virtual channels. Thus, we have support for eight virtual channels and our main requirement for LASH routing is satisfied.

B. Flow Control

Ethernet supports a variant of *on/off* flow control as opposed to *credit-based* flow control found in technologies such as Advanced Switching Interconnect [1], InfiniBand [2] and Myrinet [3]. On/off flow control consists of on/off messages, called *pause frames* (Figure 1(b)). When a downstream node has available buffer space it sends an *on* message to the upstream node indicating it is ok to transmit frames. Otherwise, it sends an *off* message telling the upstream node to halt transmission. A major drawback of Ethernet flow control is that it only allows per-port flow control. This makes it impossible for flow control on a per-priority basis. Not only does this limit the performance of the priority scheme itself [7], but it makes it impossible to support virtual channels in combination with flow control. When flow control is enabled, the virtual channels no longer have independent buffering resources.

C. *LASH Applied to Ethernet*

Ethernet does not explicitly support the virtual channels that LASH requires, but we will, as described above, use the eight priority levels as eight virtual channels. Unfortunately, due to the lack of per priority flow control, priorities can only be used as virtual channels when flow control is disabled. To remedy this we adopt a proposal, suggested by the IEEE 802.3ar Congestion Management Task Force[1], to change the Ethernet flow control mechanism to allow for per-priority flow control. We need to change the granularity of the pause frame in order to achieve per-priority flow control. The pause frame must include a field which tells the receiver what priority it should pause.

The introduction of a new pause frame is done by creating a new *OpCode* to distinguish between pause frames. The new frame will contain the same information as the old one (Figure 1(b), but with a different OpCode (Figure 1(c)) and with the priority value embedded after the pause time. This makes it possible to extract the priority constraint for a frame by reading the priority value. Per-priority aware switches will use the OpCode field to decide the type of frame received and will then scan for the necessary information accordingly. Per-port aware switches will use frames with $OpCode = 1$ as before, while frames with $OpCode = 2$ will be ignored. As aforementioned, this change is currently considered by the IEEE 802.3ar Congestion Management Task Force, but with the goal of improving congestion management.

The priority information in the pause frame enables the concept of virtual channels in combination with flow control. The STP can now be replaced (or extended, see Section IV-D) by LASH. Such a replacement can be either centralised or distributed in nature. In the centralised approach, which is similar to the way STP behaves today, all switches will go through an election process where a *master* is selected based on identity (or another property). The master switch will then build and maintain a topology map based on information collected from all other switches. The topology map is used by the master switch to calculate and distribute routing tables. After calculation these routing tables are considered fixed until a topology change occurs and recalculation is triggered, which is identical to how STP and TBTP function. In the distributed case all switches will have to collect topology information and calculate its own routing table in a deterministic fashion (i.e. all switches must make the same decisions or will will run into trouble). This process will then be repeated whenever the topology changes.

As described in Section III-B1 LASH needs a modest number of layers. A network with 16 switches will need 2 layers for routing, leaving us with 6 layers to use for other purposes such as QoS. E.g. if we have a network with 16 switches we can have a maximum of three classes of service, using a total of six layers. Two layers are required for LASH routing, but as we need separate queues for each of the classes of service we end up with a total of six layers.

[1]http://www.ieee802.org/3/ar/

D. *Backwards Compatibility*

To provide a convenient upgrade path LASH enabled switches should be backwards compatible with older switches. This should enable a gradual upgrade of network equipment, where an *island* of LASH enabled switches can work together with switches not supporting LASH. Backwards compatibility can be achieved by requiring new switches to support both STP and LASH, and dedicating virtual channel 0 to STP. How frames are routed will then be determined by the type of switch the source and destination refers to. LASH switches will route according to LASH for all LASH destinations within its island, and according to STP for all other destinations. STP switches will route according to STP for all destinations.

V. Performance Evaluation

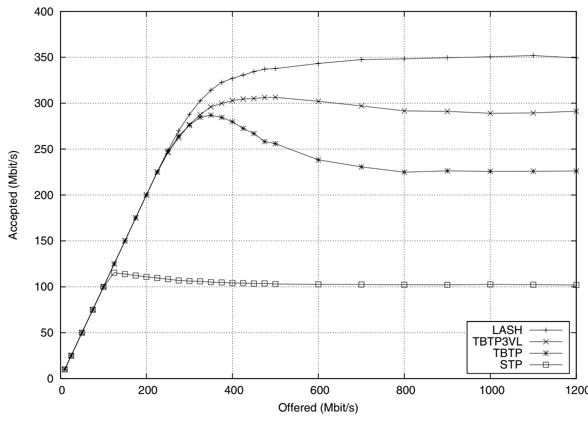
Our evaluation is based on the J-Sim framework[21], which is used to implement a shared memory Ethernet switch that is the building block of our networks. Our switches have five ports, where one is connected to a computing node and four are connected to the network. We have simulated both irregular and regular networks with a size of 16 (not included) and 32 switches. For each set of networks we have used the average throughput and latency as the performance measure. We have used a uniform destination distribution and a packet arrival process governed by a normal approximation of the Poisson distribution[2]. The average bit rate is parametrised and 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.

A. *Throughput*

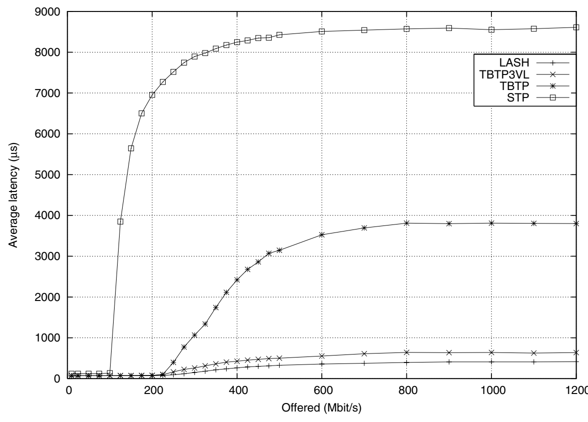
In order to have a fair comparison between LASH and TBTP we have included results where the TBTP algorithm is allowed to use the same number of channels as LASH. This means that the network traffic is spread over several layers instead of one. These results are labelled TBTP3VL in Figure 2. Results for STP are included as a reference.

Throughput results where generated for irregular networks with 32 switches, a 4x8 mesh, and a 4x8 torus. Figure 2(a), 2(c) and 2(e) summarise the average throughput for the different routing algorithms. It is evident that LASH achieves the best performance; it outperforms both TBTP and STP because all links are used, no turns are forbidden and all frames are routed via shortest paths. We see an average increase in throughput of 16% compared to TBTP3VL for irregular networks (Figure 2(a)). For the 4x8 torus the increase is 25% (Figure 2(e)) and for the 4x8 mesh an immense 83% (Figure 2(c)). For TBTP3VL the improvement over TBTP varies as the improvement yielded by more available layers depends on the topology. An improvement of 25% and 20% is achieved for irregular networks and the torus respectively. On the 4x8 mesh the addition of more layers does not benefit TBTP. The addition of further layers gives TBTP some room

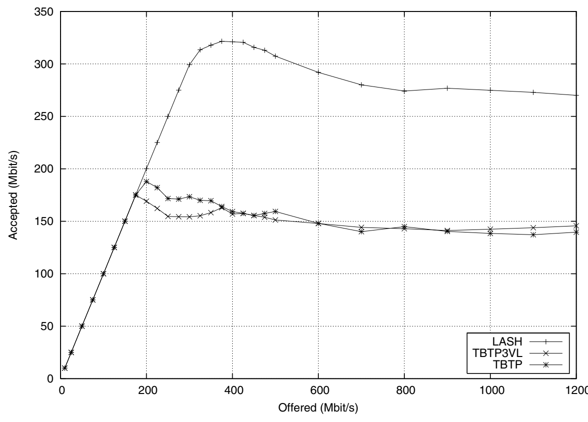
[2]Simulations for other destination distributions and arrival processes have yielded similar results.



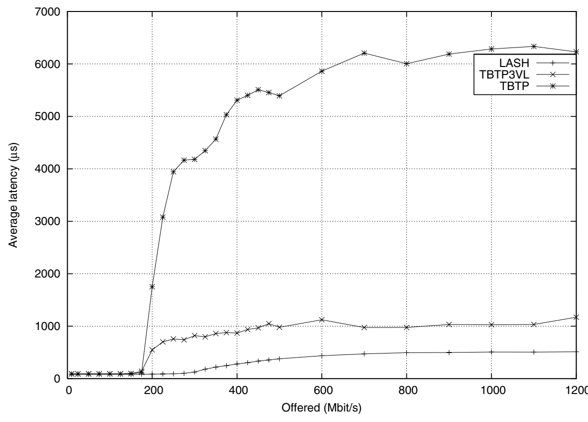
(a) Throughput for irregular topologies with 32 switches



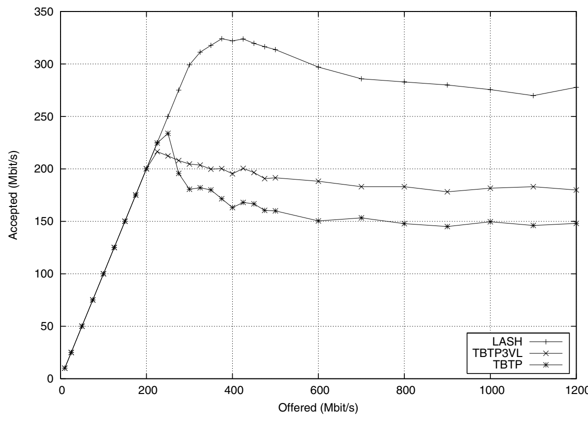
(b) Latency for irregular topologies with 32 switches



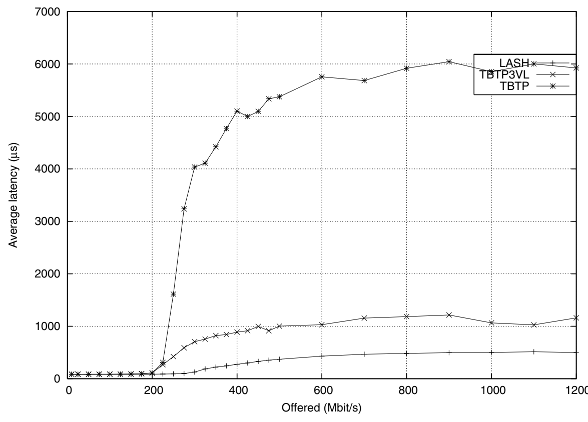
(c) Throughput for 8x4 mesh



(d) Latency for 8x4 mesh



(e) Throughput for 8x4 torus



(f) Latency for 8x4 torus

Fig. 2. Throughput and latency.

for improvement, but as the number of turns allowed and the available paths are still the same the performance increase is limited.

B. Latency

Figure 2(b) shows the average network latency for irregular topologies. As can been seems from the figure, LASH improves latency when compared to the other alternatives. LASH has an average latency of 450μs which is a 25% reduction compared to the 600μs of TBTP3VL, and a 89% reduction of that of TBTP. The latency plots for the torus and mesh can be summarised as follows. Latency is reduced by 55% between LASH and TBTP3VL for the torus, and by 52% for the mesh. No turn prohibition and shortest path routing are the reason for these improvements for LASH. For TBTP3VL the addition of more layers leads to a reduction in latency compared to only one layer. This is due to less head of line blocking when several layers are available.

VI. Conclusion

We have described how to make virtual layers a reality in Ethernet through a simple modification of the pause frame. This modification makes it possible to use topology agnostic, deadlock free and shortest path routing. Our proposed algorithm improves throughput by more than 300% over that of Ethernet with STP, and by more than 83% when compared to other routing algorithms suggested for Ethernet. Bringing per priority flow control and shortest path routing to Ethernet gives it routing performance on par with highly specialised network technologies; performance that is needed if Ethernet is to succeed in the SAN and backplane area.

References

[1] *Advanced Switching Core Architecture Specification*, revision 1.1 ed., Advanced Switching Interconnect Special Interest Group, August 2004.

[2] *Infiniband architecture specification*, 1st ed., InfiniBand Trade Association, October 2004.

[3] 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.

[4] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, *Internet Small Computer Systems Interface (iSCSI)*, rfc3720 ed., IETF, April 2004.

[5] S. Hopkins and B. Coile, *AoE (ATA over Ethernet)*, The Brantley Coile Company, Inc., August 2004.

[6] O. Feuser and A. Wenzel, "On the effects of the ieee 802.3x flow control in full-duplex ethernet lans," in *Proceedings of the 24th Conference on Local Computer Networks*, October 1999.

[7] W. Noureddine and F. Tobagi, "Selective back-pressure in switched ethernet lans," in *Proceedings of GLOBECOMM*, December 1999.

[8] S.-A. Reinemo and T. Skeie, "Ethernet as a lossless deadlock free system area network," in *Parallel and Distributed Processing and Applications: Third International Symposium, ISPA 2005, Nanjing, China, November 2-5, 2005. Proceedings*, ser. Lecture Notes in Computer Science, Y. Pan, D. Chen, M. Guo, J. Cao, and J. J. Dongarra, Eds., vol. 3758. Springer Berlin/Heidelberg, October 2005, pp. 901–914.

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

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

[11] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transaction on Computers*, vol. C-36, no. 5, pp. 547–543, May 1987.

[12] 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 AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, March 2004, pp. 2175–2184.

[13] T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson, "Smartbridge: a scalable bridge architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 205–216, 2000.

[14] K.-S. Lui, W. C. Lee, and K. Nahrstedt, "Star: a transparent spanning tree bridge protocol with alternate routing," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 33–46, 2002.

[15] R. Garca, J. Duato, and J. Serrano, "A new transparent bridge protocol for lan internetworking using topologies with active loops," in *ICPP '98: Proceedings of the 1998 International Conference on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 1998, pp. 295–303.

[16] 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. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, March 2004, pp. 2283–2294.

[17] 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*, vol. 9, no. 8, October 1991.

[18] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *J. ACM*, vol. 41, no. 5, pp. 874–902, 1994.

[19] O. Lysne, T. Skeie, S.-A. Reinemo, and I. Theiss, "Layered routing in irregular networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 1, pp. 51–65, 2006.

[20] M. Koibuchi, A. Jouraku, and H. Amano, "The impact of path selection algorithm of adaptive routing for implementing deterministic routing," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, H. R. Arabnia, Ed., vol. 3. CSREA Press, June 2002, pp. 1431–1437.

[21] H. ying Tyan, "Design, realization, and evaluation of component-based compsitional software architecture for network simulation," Ph.D. dissertation, Ohio State University, 2002. [Online]. Available: www.j-sim.org