Path-Moose: A Scalable All-Path Bridging Protocol

Guillermo IBÁÑEZ[†a)], Iván MARSÁ-MAESTRE[†], Miguel A. LÓPEZ-CARMONA[†],
Ignacio PÉREZ-IBÁÑEZ[†], *Nonmembers*, Jun TANAKA[††], *Member*, *and* Jon CROWCROFT[†††], *Nonmember*

SUMMARY This paper describes Path-Moose, a scalable tree-based shortest path bridging protocol. Both ARP-Path and Path-Moose protocols belong to a new category of bridges that we name All-path, because all paths of the network are explored simultaneously with a broadcast frame distributed over all network links to find a path or set a multicast tree. Path-Moose employs the ARP-based low latency routing mechanism of the ARP-Path protocol on a bridge basis instead of a per-single-host basis. This increases scalability by reducing forwarding table entries at core bridges by a factor of fifteen times for big data center networks and achieves a faster reconfiguration by an approximate factor of ten. Reconfiguration time is significantly shorter than ARP-Path (zero in many cases) because, due to the sharing of network paths by the hosts connected to same edge bridges, when a host needs the path it has already been recovered by another user of the path. Evaluation through simulations shows protocol correctness and confirms the theoretical evaluation results.

key words: *Ethernet, routing bridges, shortest path bridges, data centers*

1. Introduction

Ethernet switched networks are today the indisputable solution for local, campus and metropolitan networks. Ethernet offers an excellent price/performance ratio, with high compatibility between elements and simpler configuration than IP. The main problem with these networks relies on the performance and restrictions of active topology derived from the use of Spanning Tree [1]. There are standards under elaboration to avoid these limitations, like Shortest Path Bridging (SPB) [2] and RBridges [3], but these protocols are based on a link-state routing operating at layer two, which leads to more complex control and computation, needing also additional mechanisms to avoid loops. In this situation, simple, zero configuration protocols which removed the limitations of spanning tree protocol might have wide applicability in small and medium size campus and enterprise networks.

ARP-Path Switches have emerged [4] as a pure bridging approach to overcome the severe restrictions of spanning tree protocols, with a simple path set up mechanism (based on the detection of the first arrival port of ARP packets at bridge ports), with low latencies and without any configuration requirement. ARP-Path protocol has been implemented successfully and is now evolving and diversifying based on the experience obtained from implementations and simulations. The first version of the protocol and its implementations ([4]–[7] and the closely related [8]) demonstrated protocol robustness and excellent latency and throughput results. However, networks of medium and big size require increased protocol scalability. This means that forwarding tables at core bridges must be reduced. The most direct way to reduce forwarding table sizes is to set up paths between bridges instead of between hosts and thus make table sizes proportional to the number of bridges B instead of to the number of hosts H.

In this paper we describe and evaluate a tree-based protocol based on ARP-Path mechanisms and MOOSE hierarchical addressing [9] to improve the scalability of ARP-path.

The paper is organized as follows. The next section describes the base protocol. Section 3 presents the Path-Moose protocol, Sect. 4 concerns the evaluation, and Sect. 5 describes the related work. The last section summarizes our conclusions.

2. ARP-Path Base Protocol

Path setup in ARP Path protocol [3] is performed by fully flooding the standard ARP Request frame sent by the source host, snooping it at every bridge and so selecting the lowest latency path found by the ARP Request. The path in the opposite direction is set up by snooping the ARP Reply frame. Figure 1 shows a network of ARP Path bridges connected by

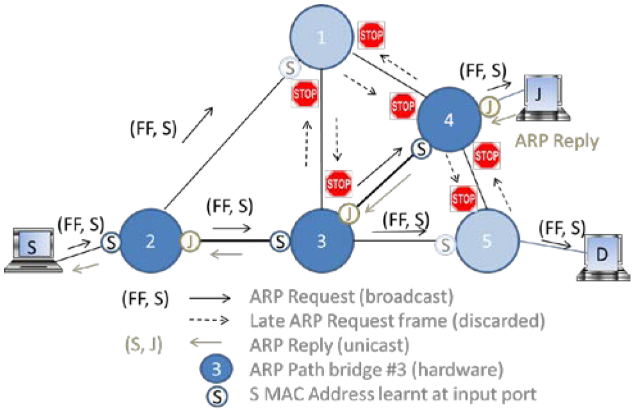


Fig. 1 Path set up between hosts with ARP path protocol with universal MAC addresses.

[†]The authors are with Universidad de Alcala, 28805 Spain.
[††]The author is with Fujitsu Ltd., Kawasaki-shi, 211-8588 Japan.
[†††]The author is with Cambridge Computer Laboratory, UK.
a) E-mail: guillermo.ibanez@uah.es

point to point Ethernet links. Host S sends an ARP Request packet encapsulated into a broadcast frame to resolve the IP address of destination host J. Every bridge forwards to all ports except the one through which it was first received and associates the global MAC address of S to this port, temporarily locking the learning of S address to this port and blocking all other ports from learning and forwarding further received broadcast frames from source address S. We illustrate the association of the source address S to this port of the bridge with a white circle and the discarding of the duplicated frames received at other ports with a “stop” signal.

When the ARP Request frame reaches host J, it responds with a unicast ARP Reply towards S that when snooped at every bridge (4,3,2) provokes the learning of source address of J at receiving port and the renewal of the learnt address S. Addresses learnt at other bridges (in light blue) that are not renewed will expire.

3. Path-Moose Protocol Description

Path-Moose protocol uses the same source address learning and locking mechanism of ARP-Path protocol, but address learning occurs with bridge granularity instead of host granularity, thus increasing scalability by reducing the size of the stored state at bridges. With Moose-based addressing, bridges learn only the bridge ID part of the source MAC address, instead of the complete host address. In this way multiple trees, each one rooted at every edge bridge, are created and refreshed by the standard ARP Requests issued by the normal host traffic. Forwarding is tree based (sink trees for unicast traffic, source trees for multicast traffic). Paths are not forced to be congruent in both directions to keep protocol simple. We define an edge bridge as the bridge connected to one or several hosts and/or to other standard bridges and core bridge a bridge connected only to other Path-Moose bridges. The edge bridges learn their host’s universal MAC addresses and translate it to a local hierarchical address of the format *bridgeID:hostID* whilst bridge ID is learnt at core bridges to build the trees. We first discuss the assignment of bridge ID to bridges; we then describe path set up and finally path recovery mechanisms.

3.1 Bridge ID Assignment

Upon network initialization, every bridge gets assigned a unique 3-byte bridge identifier (ID), as used in MOOSE. The assignment of bridge identifiers to every bridge in the network can be performed in different ways and is fully independent of the Path-Moose protocol. The specific process used to assign it is out of the scope of this document.

3.2 Host Links and Bridge Links

Path-Moose bridges, like ARP-path bridges, require point-to-point links between bridges. They may share a link between several hosts but they cannot share links between bridges because, if an output port of a bridge has a shared connection with an input port associated to a source address, replicated frames with this source address will be reinjected at that input port and will create a loop. A link can be a *host* link, connected to one or multiple hosts (or even to standard bridges), or a *bridge* link, connected (point to point) to another Path-Moose bridge. Path-Moose bridges identify the bridge links by emitting periodic *Hello* messages over all links. Links which receive Hello messages are tagged as bridge links and the others are regarded as host links. The key idea of the address learning-locking mechanism of Path-Moose bridges is that they learn (universal) host MAC addresses at host links (edge bridges) and only *bridgeIDs* at bridge links (core bridges).

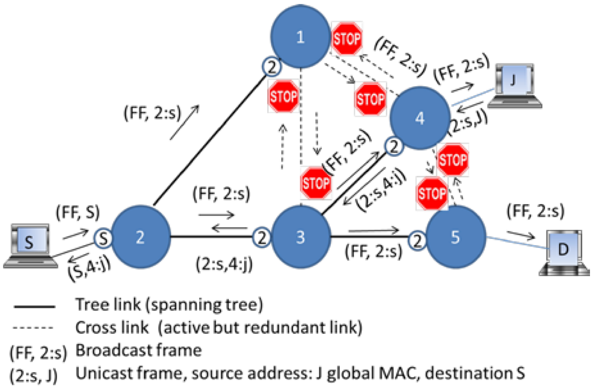

1
2
3
4
5
S
J
D
STOP
(FF, S)
(S,4:j)
(FF, 2:s)
(2:s,4:j)
(2:s,J)
2
S
— Tree link (spanning tree)
------ Cross link (active but redundant link)
(FF, 2:s) Broadcast frame
(2:s, J) Unicast frame, source address: J global MAC, destination S

Fig. 2 Setting a path and a tree rooted at bridge 2 with ARP Request from S to J and ARP Reply (J,S) learning BridgeID 2.

3.3 Sink Trees Set up between Bridges with ARP Packets

Figure 2 illustrates the process of building a path to a destination host and at the same time, without additional processing effort, building the complete sink tree rooted at the originating bridge, a tree that can route any packet towards any host connected to that bridge. Host S sends an ARP Request packet encapsulated in a frame to resolve the IP destination address. ARP Request frame arrives to its edge bridge 2. The incoming frame to the bridge gets its source address SA extracted and learnt by the bridge into a Host Table, and a local HostID address is assigned. The universal source MAC address of the frame is replaced at edge bridges by the local address obtained according to the Moose format *BridgeID:hostID* (shown as 2:*s* in figure) and placed also inside the ARP Request packet (in this way the destination host will learn the local address instead of the universal one at its ARP cache). The local address has the U/L bit set to Local indicating a private MAC address. Bridge 2 broadcasts this ARP Request frame with (2:*s*) source address over all ports except the input port. Bridges 1 and 3 receive the frame and associate (or refresh the association of) the first arrival port to the bridge ID 2, and broadcast it over all ports except the input port. Later frames arrive at bridges 1 and 3 from 3 and 1 respectively and are discarded directly because their source address is associated to another port. The ARP

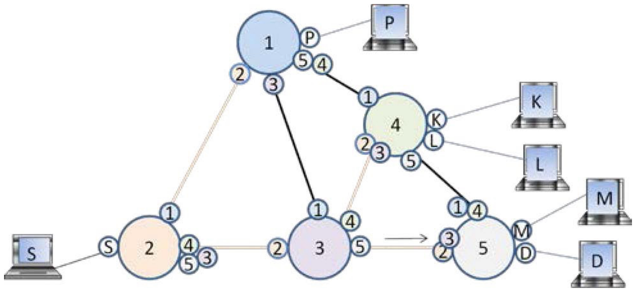


Fig. 3 Built trees. Bridge IDs and host addresses learnt after traffic flown. Tree rooted at bridge 2 is shown in orange.

Request finally arrives to the edge bridges and end hosts. The destination host J learns the local address of S (2:*s*) at its ARP cache and responds with an ARP Reply frame with its universal MAC address J as source address inside the ARP payload packet. Edge bridge 4 replaces, like in the MOOSE protocol, the universal MAC address J by the hierarchical format 4:*j* in the ARP Reply packet, forwards it via the port associated to *bridgeID 2,* and associates the input port to *bridgeID 4* if it does not have already another address assigned. The frame arrives to intermediate bridge 3 where it is forwarded via the port associated to bridge 2.

When the frame arrives at bridge 2, the hierarchical destination address is replaced by the universal table stored at the bridge host table *(HoT)* and delivered to the host. This host learns at his ARP cache the local address of J (4:*j*) instead of the universal address. From now on, the path is established in both directions.

Figure 3 shows the bridge IDs learnt by all bridges after some traffic has been sent over the network. Note that host addresses are learnt at host ports of edge bridges.

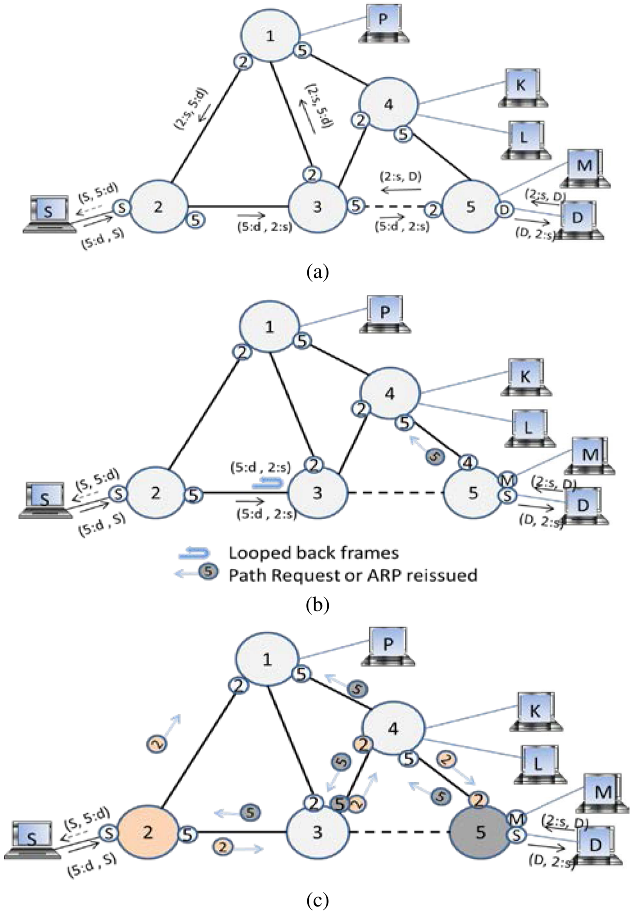
3.4 Path Recovery

Path recovery after link failure is lighter and much faster on average than in the ARP Path protocol. The reason is the sharing of paths by all hosts connected to the same edge bridge (ARP path protocol may share only paths arriving to the same host).

If a link or bridge fails, the ports connected to that failure point will detect it and delete all the table entries (*bridgeIDs*) associated to that port in the forwarding table. When a unicast frame arrives to any of the two bridges that detected the link failure and finds out there is no path to reach the destination, the path recovery mechanism is started for that destination *bridgeID*. It will often happen that normal broadcast packets sent through the destination bridge in the opposite direction will create a tree towards the destination bridge, thus accelerating path recovery and making it redundant, as described in simulation results.

The path recovery mechanism is shown in Fig. 4. There is a flow from host S to D through the lower path (bridges 2-3-5) and the flow from D to S goes via bridges 5-3-1-2.

After link 3-5 fails, bridge IDs 2 and 5 are flushed from the forwarding tables of bridges 5 and 3 respectively.



(a)

(b)

(c)

Fig. 4 Path recovery (a) Failure of link 3-5. (b) loopback of frames at 3 and Path Request/ARP at 3 (c) Path requests from 2 and 5 to restore the trees.

Frames from S, with source address (2:*s*) arriving at 3 have destination address unknown at bridge 3. The first frame is looped back towards the source edge bridge using their source *bridgeID* to route it towards bridge 2, and a short repair timer is triggered so that further frames with same destination bridge (5) are discarded to prevent further repetitive path recovery attempts. The looped back frame arrives to bridge 2 and is detected by bridge 2 as such because it arrives at the port associated with the destination bridge. Bridge 2 detects that is the source edge bridge and sends a broadcast Path Request packet with encapsulated destination address (5:*d*) (or an ARP Request from 2:*s* to resolve MAC of IP address D),which is forwarded through all network and rebuilds at the same time the tree rooted at bridge 2.

The same path recovery process happens at bridge 5 for frames with destinations to (2:*). When the link 3-5 becomes available again, the next ARP Request from a host may select the link 3-5 as the path towards 5, if the branch has a lower latency. The same occurs for the bridge 2 tree.

4. Evaluation

We evaluated the performance of the Path-Moose protocol,

Table 1 Network data for comparison Path-Moose versus ARP path (I).

Topology	Host per edge bridge *He*	Total hosts *H*	Edge bridges *Eb*	Total bridges *B*	Avg #bridges per core path *b3*
250 host network	25	250	10	14	1
3x3 mesh 150	25	150	6	9	1.5
VL2 Clos network	20	25000	1250	1350	3

Table 2 Table sizes (entries) calculations Path-Moose versus ARP path.

	Path-Moose		ARP Path & STD		*Table*	
			bridges		*reduction*	
	Eb-1+He	*Eb*	5*H/Eb	H*b3/(B-Eb)	*ratio (times)*	
Topology	Table size edge bridges max.	**Table size core bridge**	Table size edge bridges average	**Table size core bridges**	*Ratio edge*	***Ratio core***
250 hosts network	34	**10**	89	**94**	*2,6*	***9,4***
3x3 mesh 150 hosts	30	**6**	83	**50**	*2,8*	***8,3***
VL2 Clos network	100	**33**	100	**500**	*1,0*	***15***
	5*H/B	*Da/3*	5*H/B	2*H/Da		

and compared it with its predecessor, the host-based ARP Path protocol, and also with standard backward learning bridges. We compared forwarding table sizes because they are a key factor for network scalability. We also compared path recovery times, essential for maximum network availability. We first evaluated the protocol with analytical estimations and then through simulations. As a reference, the original ARP Path protocol itself was compared with Shortest Path Bridges and Spanning Tree bridges in [4], [5].

4.1 Theoretical Analysis. Number of Table Entries

Table 1 and Table 2 show, respectively, the network parameters and the calculation results for three networks: a 250 host network (Fig. 5) with four core bridges and 10 edge bridges (25 hosts connected to every edge bridge), a 3 × 3 regular mesh network with 150 hosts (not shown) with 25 hosts connected to every bridge at the left and right columns, and a non-blocking VL2 intermediate Clos network (Fig. 6) as proposed in [12] for big data centers. Table 1 shows network parameters. Table 2 compares the estimated table sizes of Path Moose and ARP path bridges. Formulas used for the first two networks used are shown on top of columns. Formulas used for the VL2 network are shown in the bottom line correspond to the VL2 topology.

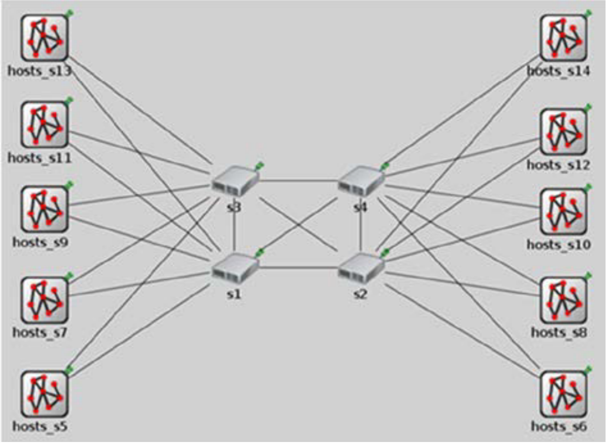


Fig. 5 Network of 250 hosts (DC250) The ten subnetworks of 25 host connected to access switch *X* are shown (collapsed) as *hosts_sx*.

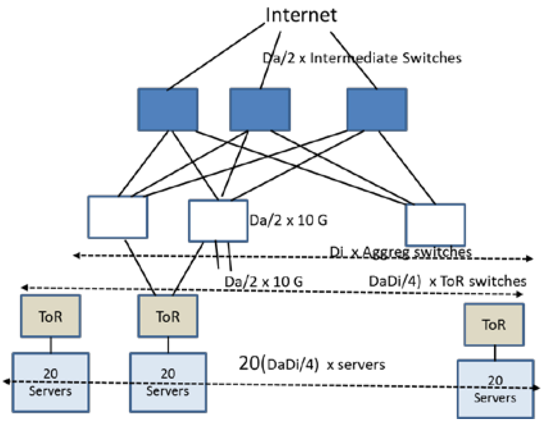


Fig. 6 Clos intermediate network for data center [12].

4.1.1 Path-Moose Bridges

For the first two topologies, the maximum forwarding table size at edge bridges is *Eb-1+He*, where *Eb* is the number of edge bridges and *He* is the number of (active) hosts connected per edge bridge. This expression for table size at Path-Moose edge bridges results from the fact that there will be an active entry per edge bridge (minus the bridge itself) and one per each active directly connected host, with an average maximum of *He=H/Eb* active hosts per bridge.

Most edge bridges will likely be active because the probability for an edge bridge of having at least one active host is very high, considering a typical range of 20–80 connected hosts per bridge. Additionally, edge bridges keep a table of directly connected hosts (*host table*) containing its universal and local addresses and associated bridge port. This is a consequence of the fact that, although bridge IDs are learnt at core ports, host addresses are learned and translated at edge bridges, thus reducing the theoretical reduction of stored state at edge bridges. At core bridges the size of forwarding tables equals *Eb* (the number of edge bridges) because one entry per edge bridge is enough to obtain complete routes to any network host.

4.1.2 Standard Bridges and ARP Path Bridges

Both standard 802.1D transparent learning bridges and ARP path bridges behave similarly regarding sizes of forwarding tables. Although there are some differences in address learning, the effect in table size is small and only applies to reconfiguration situations. The main difference is that standard bridges replicate the frames with unknown unicast addresses, whilst ARP path bridges trigger a path repair sending some broadcast packets that provoke destination (and source) address learning and finally repair the path. This process takes little time and occurs only upon link or bridge failure.

The table sizes of standard and ARP path edge bridges depend on the number of active flows per edge bridge. This may vary widely depending on the traffic matrix characteristics, including its locality properties, sometimes enhanced by network administrators to optimize network performance. We take from the measurements in [12] a ratio of 5 simultaneous active flows per host as a moderate average. A high locality in the traffic will tend to place flows in the nearest switches, reducing table sizes.

Core bridges are the bottleneck for standard bridges and ARP Path bridges regarding forwarding table sizes because they concentrate most network traffic over a small set of core bridges. In the DC250 core (250 host) network, 24 bridge topology, the four core bridges learn all host addresses. The maximum total number of entries at network core bridges can be expressed by $Te = H * b3/(B\text{-}Eb)$, where $b3$ is the number of core bridges traversed in the average network path at core and *B-Eb* is the total number of core bridges. In the above-mentioned network (DC250) at Fig. 4, it is easy to calculate $b3 = 1,5$ bridges.

4.1.3 Clos Network

The Clos intermediate non-blocking network for data center proposed in [12] is representative of recent data center topologies. We use it here for an estimation of table sizes in big networks. Note that our DC250 host data center topology is a small subset of this topology (ToR switches being the access switches) if vertical links of DC250 are eliminated (they carry negligible traffic). Each aggregation switch has *Da/2* links. There are *Di* aggregation switches, *Da/2* intermediate switches and *Da*Di/4* Top of Rack (ToR) edge switches as shown in Fig. 6. It has a bisection bandwidth of *5*Da*Di* Gbps, with 20 servers per ToR switch with 1 Gb links. For the Clos network calculations we assume *Da=100* and *Di=50*, with a total of 25.000 servers. The formulas used for Path-Moose are as follows: for edge bridges, table sizes are similar in Path-Moose and standard bridges because the load is widely distributed among the intermediate switches due to the high network size and highly connected network topology. Due to the high network sizes and low host per edge bridge ratio, ToR switches will not have inputs at their tables for all edge bridges.

The size ratios at edge bridges are modest and even there may be no reduction in size as shown for the arbitrary big networks. Table size is not critical for edge bridges, but it is for core bridges, due to their strict performance requirements.

When *He* is low and *Eb* is high, the edge bridge tables are similar in standard and Path-Moose bridges and depend on the number of active flows per host [12]. The use of the Path-Moose protocol in the Clos network achieves, compared with ARP path protocol and standard bridges a reduction in number of table entries of up to 20 times at core bridges. We assume an average factor of *Da/3* (between maximum of *Da/2* and a minimum *Da/4*) for minimum spreading of ToR traffic over diverse intermediate switches.

4.2 OMNeT++ Simulations

The protocol has been implemented in OMNeT++ INET simulation environment [11]. Both path and tree creation and path recovery were successfully tested in different topologies. Trees, paths, and traffic flows were created without frame loops. To create realistic traffic loads, a flow generator was implemented in OMNET++. The flow model is based on [16] and [17]: there is a single flow generator that sequentially produces new flows with exponentially distributed inter-arrival times. The inter-arrival time is configured to control the average traffic intensity. The traffic model is inspired in [2], with flow sizes with Pareto distribution (α=1,3) and three transfer rates: 30% of flows at 0.5 Mbps, 60% of flows at 1 Mbps and 10% at 10 Mbps. The flow source and destinations are selected at random among all hosts, with equal probability. The flow is divided in sequential packets of 500 bytes. Packet size does not affect the protocol because during the time the flow is active the path is refreshed and does not change. Flow rate neither affects address learning.

For this test, we used the topology shown in Fig. 5, consisting of 4 core switches and 10 access bridges. Link speed is 100 Mbps in all links and link delay is 1 microsecond per link. This network has 25 hosts connected to each of the 10 edge bridges as shown in the figure, which are connected to four fully interconnected core bridges. The simulated topology change consists of a failing link between switches s1 and s2 (one of the essential links).

Tests on this network with ARP-path and Path-Moose protocol have been carried out by sending information between hosts using the ping application and UDP traffic with the above described traffic generator. The main measurements are path setup and recovery times and forwarding table sizes of bridges.

Table 3 Path setup and path recovery times of ARP Path vs. Path-Moose.

	Path set up		**Path recovery**	
Seconds	ARP-path	*Path-Moose*	**ARP-path**	***Path-Moose***
Average	4.32E-05	*4.32E-05*	**3,42E-05**	***3,64E-06***

4.2.1 Path Setup and Path Recovery Times

Path setup and recovery times were measured for ARP Path and Path-Moose protocols. Path setup times are identical because the mechanism to set up the path is the same. Path recovery times of Path-Moose equal also ARP-path path recovery times except in the cases where two hosts (H1 and H2) connected to the same edge bridge (B1) having a communication with other two hosts (H3 and H4) both connected to other edge bridge (B2). In this case, path recovery delay with Path-Moose is zero for all flows from the second flow onwards (H1 ↔ H3 or H2 ↔ H4) because paths are set up between border bridges instead of independently per host, so restoring one path between edge bridges is enough to restore all paths between all hosts sharing the same path. This reduces greatly the average path recovery time by a factor of ten in our measurements and depends on the traffic distribution over the failed link. Results are shown in Table 3.

It is worth noting that, although ARP Path can also use an already existing path to the same destination host for path recovery, Path-Moose is much more efficient because it can use any existing path to the same destination bridge. Path recovery is on average one order of magnitude faster that ARP-Path. This is an effect of the path aggregation obtained with Path-Moose hierarchical addresses. It does depend on the number of hosts connected per edge bridge and their traffic intensity, but not on the network topology.

4.2.2 Forwarding Table Sizes

Table sizes were measured in bytes including the size of the address translation tables at edge bridges. Results for the 250 hosts datacenter network and for the 3×3 mesh network are shown at Table 4.

In the core bridges and in most edge bridges, Path-Moose requires less space for storing routing information. Reduction ratio is very important at core bridges, those crucial for network scalability. The reason is that Path-Moose stores only the *bridgeID* and port (two bytes) in the forwarding tables, instead of the full MAC address plus the port (7 bytes) and that *bridgeID* is shared by multiple paths. For core bridges ratios of table sizes of 16 and 24 (more than double than the theoretical ratio for table lengths) is obtained, because not only we have less entries, but also the entry width is halved. Big networks might need three bytes per entry to store bridgeID and port, reducing the sizes ratio. The key parameter however, is maximum table length at core bridges.

Table 4 Forwarding table sizes (bytes).

	Path-Moose *Table sizes*		ARP Path bridges *Table sizes*		*Table reduction ratio (times)*	
Topology	Edge bridges	**Core bridges**	Edge bridges	**Core bridges**	*Edge bridges*	***Core bridges***
250 hosts network	136	**20**	277	**325**	*2,0*	***16***
3x3 mesh 150 hosts	139	**11,3**	331	**274**	*2,4*	***24,2***

Table 5 Forwarding table sizes (number of entries).

	Path-Moose *Table sizes*		ARP Path bridges *Table sizes*	
Topology	Edge bridges	**Core bridges**	Edge bridges	**Core bridges**
250 hosts network	9 (21)	**10**	39,7	**55,9**
3x3 mesh 150 hosts	5 (18)	**6**	67,1	**61**

Results at Table 5 compare table sizes in number of entries obtained with the simulator with moderate traffic (flow inter arrival time of 0.08 sec). At edge bridges we show both the number of forwarding table entries (without brackets) and the total number of entries including the compacted addresses table (between brackets). The first term is constant (*Eb-1*) because there is always an active flow between every pair of edge bridges. It is worth noting that values for Path-Moose are independent of traffic intensity (once beyond low traffic threshold intensity) while values for ARP-Path bridges and standard bridges increase with traffic matrix dispersion, traffic intensity and network topology. As an example, traffic matrix at 3 × 3 mesh is from left hosts towards right hosts, increasing table sizes compared with random traffic matrix used at 250 hosts network. These are the reasons behind the variability of table sizes obtained for ARP Path bridges for the two topologies versus the nearly constant and reduced size of Path-Moose tables. Taking this into account, big data center networks like Clos networks described above, will have core bridges with tables with maximum of one entry per edge bridge with Path-Moose protocol and of one entry per host worst-case with ARP-path protocol, in other words, up to *H/Eb* times worst-case longer tables.

The conclusions of the simulation results are that path recovery with Path Moose is much simplified and many hosts get their path repaired in advance by others. Bridges learn very fast other bridges' addresses from any broadcast message, like ARP and DHCP issued by any host attached. Table sizes confirm theoretical results with significant reduction and constant size at core bridges (i.e. one entry per edge bridge) and moderate reduction at edge bridges.

5. Related Work

The two main protocol proposals currently under advanced stages of standardization at IEEE and IETF are Shortest Path Bridges (SPB) [12] and RBridges (TRILL) [13]. Both use specific adaptations of the IS-IS link state routing protocol to compute routes or shortest path trees between bridges. SPB offers two choices of data plane: SPBV (Q-in-Q) and SPBM (MAC-in-MAC). Whilst with SPBV all bridges still learn MAC addresses from frames in transit as in 802.1D, SPBM learns backbone bridge connectivity via IS-IS messages. Due to the requirements for path congruency and equal cost multipath routing, SPB has a extremely high computational complexity of $\Theta(N^3)$ as shown in [15].

On the other hand, Rbridges provide optimal pair-wise forwarding and support for multipath routing of both unicast and multicast traffic. RBridges have the advantage of being fully compatible, (not only in core-island mode as SPB and ARP path) with standard IEEE 802.1 bridges and end nodes, but this is at the high cost of increased forwarding complexity: the destination address of the next-hop RBridge must be inserted in the outer header of the frame at each RBridge.

6. Conclusion

The main advantages of Path-Moose are enhanced scalability and fast and simple reconfiguration (path recovery). Path-Moose core bridges require smaller forwarding tables (aprox. 8–15 times) than standard and ARP path bridges and its maximum table length equals the number of edge bridges. The improvement is much more significant for core switches, which are critical for protocol scalability. Further study on tree maintenance policies and reduction of broadcast traffic by applying ARP proxies or distributed directory systems will help to optimize the protocol for different network environments.

Acknowledgments

This work was supported in part by grants from Comunidad de Madrid and Universidad de Alcalá through the following projects: MEDIANET-CM (S-2009/TIC-468), EMARECE (PII1I09-0204-4319) and CMAC (UAH2011/EXP-016).

References

[1] IEEE 802.1D-2004 IEEE standard for local and metropolitan area networks 2004-Media access control (MAC) Bridges. Available online: http://ieee.org/getieee802/802.1.html

[2] Shortest Path Bridging, IEEE 802.1aq. Available at: http://www.ieee802.org/1/pages/802.1aq.html

[3] Transparent interconnection of lots of links (TRILL) WG, Available online at: http://datatracker.ietf.org/wg/trill/charter/

[4] G. Ibanez, J.A. Carral, J.M. Arco, D. Rivera, and A. Montalvo, "ARP path: ARP-based shortest path bridges," IEEE Commun. Lett., vol.15, no.7, pp.770–772, July 2011.

[5] G. Ibanez, J.A. Carral, A. Garcia-Martínez, J.M. Arco, D. Rivera, and A. Azcorra, "Fast path ethernet switching: On-demand efficient transparent bridges for data center and campus networks," 17th IEEE Workshop on LANMAN, May 2010.

[6] G. Ibanez, B. de Schuymer, J. Naous, D. Rivera, E. Rojas, and J.A. Carralm, "Implementation of ARP-path low latency bridges in Linux and OpenFlow/NetFPGA," IEEE 12th Int. Conf. High Performance Switching and Routing Conference HPSR, pp.30–35, Cartagena, July 2011.

[7] E. Rojas, J. Naous, G. Ibanez, D. Rivera, J.A. Carral, and J.M. Arco, "Implementing ARP-path low latency bridges in NetFPGA," Poster-demo at SIGCOMM, Toronto, Aug. 2011. http://dx.doi.org/10.1145/2043164.2018512

[8] K. Miyazaki, K. Nishimura, J. Tanaka, and S. Kotabe, "First-come first-served routing for the data center network: Low latency loop-free routing," World Telecommunications Congress (WTC), 2012, pp.1–6, March 2012.

[9] M. Scott and J. Crowcroft, "Addressing the scalability of Ethernet with MOOSE," http://www.cl.cam.ac.uk/˜mas90/

[10] OMNeT++ Simulator, Available on line: omnetpp.org

[11] INET Framework, Available on line: http://www.inet.omnetpp.org

[12] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," Proc. ACM SIGCOMM 2009 Conference on Data communication (SIGCOMM'09). ACM, New York, NY, USA, pp.51–62, 2009.

[13] Rbridges, Bidirectional Forwarding Detection (BFD) support for TRILL, http://wiki.tools.ietf.org/html/draft-ietf-trill-rbridge-bfd-02

[14] Norman Finn, Shortest Path Bridging, http://ieee802.org/802_tutorials/05-July/nfinn-shortest-path-bridging.pdf. 2005

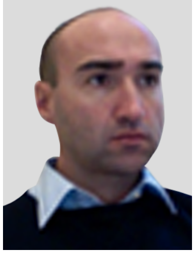
[15] J. Farkas and Z. Arato, "Performance analysis of shortest path bridging control protocols," GLOBECOM 2009.

[16] R.S. Prasad and C. Dovrolis, "Beyond the model of persistent TCP flows: Open-loop vs. closed-loop arrivals of non-persistent flows," 41st Annual Simulation Symposium (anss-41 2008), pp.121–130, 2008.

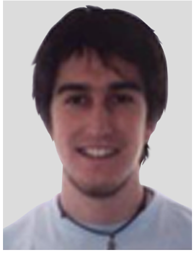
[17] A. Kvalbein, C. Dovrolis, and C. Muthu, "Multipath load-adaptive routing: Putting the emphasis on robustness and simplicity," Proc. ICNP, pp.203–212, 2009.

Guillermo Ibáñez received his Telecommunication Engineering degree from Universidad Politécnica de Madrid in 1975 and the Ph.D. in Communication Technologies from Universidad Carlos III de Madrid in 2005. He worked at IT&T R&D Labs and at Alcatel. He is an Associate Professor in the Telematics Engineering Area of the Universidad de Alcala in Madrid. He is author of multiple publications and patents on advanced Ethernet switching.

Iván Marsá-Maestre received his Telecommunication Engineering degree in 2003 and the Ph.D. from the Universidad de Alcalá in 2009. He has been working as a lecturer for the Computer Engineering Department of the Universidad de Alcalá since 2004. From his research have emerged collaborative research lines with the Center for Green Computing, at the Nagoya Institute of Technology (Japan) and the Center for Collective Intelligence, at the Massachusetts Institute of Technology (USA), where he was working as visiting researcher during the year 2010–2011.



Miguel A. López-Carmona received his B.E. in Electronics Engineering from the Universidad de Alcala (Madrid, Spain), M.E. in Telecommunication Engineering from the Polytechnic University of Madrid, and Ph.D. degree from the Universidad de Alcala in 2006. From 1995 to 2006 he worked in Logytel and Alcatel as project manager and research scientist and as assistant professor From 2006 he is associate professor at the Department of Computer Engineering at the Universidad de Alcala.



Ignacio Pérez-Ibáñez received its B.S. in Informatic Engineering in 2009 and its M.S. in 2012. He has performed research work in spanning tree protocols and advanced Ethernet networks and security. He currently works at TYPSA.

Jun Tanaka received his B.E. and M.E. of electrical engineering from Tohoku University in 1987 and 1989 respectively. Since then, he engaged in R&D of ATM transmission systems, Ethernet transmission systems and datacenter networking. His research interest includes quality of service, network virtualization, and Ethernet fabric technology. He is author of several publications and patents on advanced Ethernet protocols. He is a member of IEEE.

Jon Crowcroft has been the Marconi Professor of Communications Systems in the Computer Laboratory since October 2001. He has worked in the area of Internet support for multimedia communications for over 30 years. He leans towards a “build and learn” paradigm for research. He graduated in Physics from Trinity College, University of Cambridge in 1979, gained an MSc in Computing in 1981 and Ph.D. in 1993, both from UCL. He is a Fellow of the ACM, a Fellow of the British Computer Society, a Fellow of the IET and the Royal Academy of Engineering and a Fellow of the IEEE. He likes teaching, and has published a few books based on learning materials.