Implementing ARP-Path Low Latency Bridges in NetFPGA

Elisa Rojas[1], Jad Naous[2], Guillermo Ibañez[1], Diego Rivera[1], Juan A. Carral[1], Jose M. Arco[1]
elisa.rojas@uah.es, jnaous@gmail.com, guillermo.ibanez@uah.es, diego.rivera@uah.es, jac@aut.uah.es, josem.arco@uah.es

[1]University of Alcala
Campus Externo, Alcalá de Henares 28805, Spain

[2]Stanford University
Stanford, California 94305, USA

ABSTRACT
The demo is focused on the implementation of ARP-Path (a.k.a. FastPath) bridges, a recently proposed concept for low latency bridges. ARP-Path Bridges rely on the race between broadcast ARP Request packets, to discover the minimum latency path to the destination host. Several implementations (in Omnet++, Linux, OpenFlow, NetFPGA) have shown that ARP-Path exhibits loop-freedom, does not block links, is fully transparent to hosts and neither needs a spanning tree protocol to prevent loops nor a link state protocol to obtain low latency paths. This demo compares our hardware implementation on NetFPGA to bridges running STP, showing that ARP-Path finds lower latency paths than STP.

Categories and Subject Descriptors
C.2.5. [**Computer-Communication Networks**]: Local and Wide-Area Networks – *Ethernet*.

General Terms
Performance, Design, Experimentation,Verification.

Keywords: Ethernet, Routing bridges, NetFPGA, Shortest Path Bridges, Spanning Tree.

1. INTRODUCTION
Ethernet switched networks offer important advantages in terms of price/performance ratio, compatibility, and simple configuration without the need for IP address administration. But the spanning tree protocol (STP) [1] limits the performance and size of Ethernet networks. Current standards proposals, such as Shortest Path Bridges (SPB) [2] and Routing Bridges [3] rely on a link-state routing protocol, which operates at layer two, to obtain shortest path routes and build trees rooted at bridges. However, they have significant complexity both in terms of computation and control message exchange and need additional loop control mechanisms.

2. ARP-PATH PROTOCOL
2.1 ARP-Path Path setup
The ARP-Path protocol relies on the race between flooded ARP requests to establish the fastest path. Note that only ARP frames (or special broadcast frames in failure cases) discover or create new paths. [4]

2.1.1 ARP-Path Broadcast Discovery (ARP Request)
When host *S* wants to send an IP packet over Ethernet to host *D* over IP, it needs *D*'s MAC address. If the mapping of *D*'s IP address to *D*'s MAC address is not in *S*'s ARP cache, *S* broadcasts an ARP Request, *B*, for *D*'s MAC address (Figure 1-a). Ingress bridge 2 receives the frame from *S* and temporarily associates (locks) *S*'s MAC address to the ingress port. Unlike traditional learning switches, further broadcast frames from *S* arriving to other input ports of bridge 2 will be discarded because they arrived over slower paths. *S*'s address is now in a locked state and bridge 2 broadcasts *B* on all other ports (Figure 1-b). Bridges 1 and 3 behave similarly, locking *S*'s address to *B*'s ingress port and broadcasting *B* over all other ports, thus sending duplicate copies to each other. Because these frames arrive at a different port from the one already locked to *S*'s MAC address, they are discarded (Figure 1-c). In turn, bridges 4 and 5 process *B* the same way, finally delivering *B* to the destination host *D*. There is now a chain of bridges, each with a port locked to *S*'s MAC address forming a temporary reverse path from *D* to *S* (Figure 1-c).

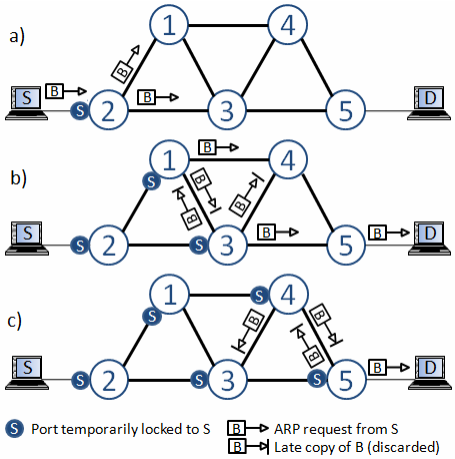

Figure 1: ARP-Path discovery from host *S* to host *D*. The small bubbles on the links show which bridge port locked *S*'s address

2.1.2 ARP-Path Unicast Discovery (ARP Reply)
The next step is in the reverse direction (i.e. from *D* to *S*) when host *D* sends the ARP Reply to host *S* in a unicast frame *U*, with *S*'s MAC address as destination address. Given the temporary reverse path back to *S* that was established by the ARP Request frame, *U* can be delivered with no further broadcasts. Like the ARP Request frame, *U* establishes a path from *S* to *D* for other unicast packets from *S* to *D*. Note that ARP-Path only establishes symmetric paths.

2.1.3 Unicast/Multicast/Broadcast communication
Once a bidirectional path is established (such as the one between *S* and *D*), all unicast frames between the two endpoints use that path. Multicast and broadcast frames use a loop-free broadcasting mechanism similar to that described for ARP Requests above.

SIGCOMM'11, August 15-19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

ARP-Path bridges only accept frames from a particular source at the port that receives the first multicast or broadcast frame from that source. Unlike ARP Requests, other multicast and broadcast frames do not establish new paths.

2.1.4 Path Repair

When a unicast frame arrives at a bridge, the bridge may not know the output port for the frame's destination MAC address. The entry could have expired, or a link or a bridge might have failed. The Path Repair protocol emulates an ARP exchange to establish a new path, using PathFail, PathRequest, and PathReply messages. PathRequest messages are similar to ARP Request frames and establish the new path to the unknown destination. Thus a full path from to the destination end-host is restored.

2.2 Advantages

The protocol has several important advantages over other protocols that build routes *a priori* before any packet transmissions.

- *Minimum Latency*: The selected path is the minimum latency path as found by the ARP Request message.
- *Zero configuration*: There is no need to configure hosts and bridges.
- *Simplicity*: Bridges mainly behave as learning switches with optional ARP proxying to reduce broadcasts.
- *Load distribution and path diversity*
- *Scalability*: ARP broadcast traffic can be reduced dramatically by implementing ARP Proxy function inside the switches as shown in [5].

3. ARP-PATH OVER NETFPGA DEMO

The objective of the NetFPGA [6] implementation is to understand the robustness and throughput of ARP-Path transparent bridges in 1 Gbit/s wired networks, without the spanning tree protocol or any ancillary routing protocol operating at layers two or three.

To test the implementation we will use a PC with four NetFPGAs installed, that will behave as separated switches, connected between them and to external hosts. A user interface was created to visualize results that will launch the scripts that run the demonstration, and will build graphs to show the latencies obtained.

3.1 ARP-Path vs. STP

The main goal is to compare ARP-Path's behaviour with that of STP. The setup will consist of one desktop with 4 NetFPGAs and 2 NICs. The connections are to be physically as indicated in Figure 2.

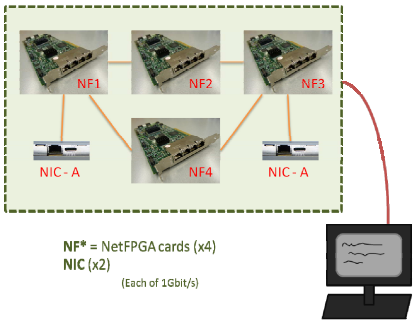

Figure 2: 4 NetFPGAs and 2 NICs connection, for the test.

The four NetFPGAs (NF1, NF2, NF3 and NF4) operate as ARP-Path bridges in one part of the demo and as STP bridges (NICs operating as separate STP bridges managed using Linux's bridge_utils) in another. We will show that ARP-Path chooses lower latency paths as opposed to STP that builds a routing tree rooted at an arbitrary switch.

3.2 ARP-Path switches Path repair

The second goal is to show that the protocol path repair is fast. Hosts *A* and *B* will start a video streaming communication. Host *A* will act as a HTTP server and *B* will connect to it and start streaming a video. We show ARP-Path's Path Repair's effectiveness after sucessive link failures and its minimal effect on the streamed video. See Figure 3.

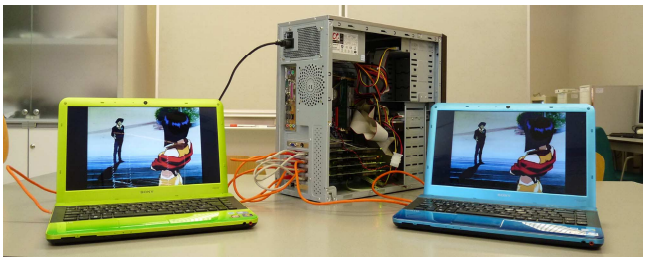

Figure 3: Two hosts connected to a PC with four NetFPGA installed

4. ACKNOWLEDGMENTS

This work was supported in part by grants from Comunidad de Madrid and Comunidad de Castilla la Mancha through Projects MEDIANET-CM (S-2009/TIC-1468) and EMARECE (PII1I09-0204-4319).

5. REFERENCES

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

[2] Allan, D., Ashwood-Smith, P., Bragg, N., Farkas, J., Fedyk, D., Ouellete, M., Seaman, M., and Unbehagen, P. Shortest path bridging: Efficient control of large Ethernet networks. *Communications Magazine, IEEE*, 48(10):128-135, October 2010.

[3] Transparent interconnection of lots of links (TRILL) WG. https://datatracker.ietf.org/wg/trill/charter/.

[4] Ibañez, G., Carral, J. A., Martinez, A. G., Arco J. M., Rivera, D. and Azrorra, A. Fastpath Ethernet switching: On-demand efficient transparent bridges for data center and campus networks. LANMAN, May 2010. Available on-line at: hdl.handle.net/10017/6298.

[5] Elmeleegy, K. and Cox, A. EtherProxy: Scaling the Ethernet by suppressing broadcast traffic. *Proceedings of IEEE INFOCOM,* 2009, Rio de Janeiro, Brazil.

[6] NetFPGA: http://www.netfpga.org