A Small Data Center Network of ARP-Path Bridges made of Openflow Switches

Guillermo Ibáñez, Jad Naous, Elisa Rojas, Diego Rivera, Bart De Schuymer, Thomas Dietz

***Abstract*—This demo shows, in a small data center demo network topology the ARP Path bridges functionality and robustness over varied platforms and their interoperability. ARP-Path bridges set up on-demand shortest paths between hosts using the standard ARP Request, but flooded over all links, to find the lowest latency path to the destination host. ARP Path is loop free, uses the standard Ethernet frame and compatible with hosts and routers.Data center demo network consists of ten ARP Path bridges. Eight bridges are implemented with standard Openflow switches (four NEC PF8800/PF5240 Open flow capable switches and four Soekris boxes running Open flow Switch implementation) and the other two are implemented on OpenWRT. A previous demo showed (in remote lab) the protocol operation over four NetFPGA boards executing Openflow[7]. Demo shows network operation with standard applications like video, demonstrating the protocol robustness, zero configuration, fast network reconfiguration upon link failures and after mobility of a host.. Implementations in Linux and Openflow show inherent robustness and fast reconfiguration. Additionally, information and video of the Sigcomm 2011 demo with implementation of ARP Path protocol on a network of four NetFPGA boards (pure hardware implementation) is also posted [8].**

***Index Terms*—Ethernet, Routing bridges, Shortest Path Bridges, Spanning Tree**

I. Introduction

Ethernet switched networks offer enormous advantages in terms of price/performance ratio, compatibility and simple configuration. Unfortunately, the spanning tree protocol (STP) [1] limits severely the performance and size of Ethernet networks. Current standards under discussion like 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. Both have significant complexity in terms of computation and control message exchange and need additional loop control mechanisms. In this paper we describe a demo of a fully operating network of ARP-Path Bridges [4]. ARP-Path is a simple, *all-path*, low latency, zero-configuration bridging protocol suitable for metro, campus, enterprise, and data center networks that enables the use of all available links without performing routing computations or a spanning tree. Simulations show low latency, full infrastructure utilization and similar throughput and delay than shortest path routing at a fraction of its complexity.

II. ARP-Path Protocol

A. ARP-Path Path setup

The ARP-Path protocol uses ARP Requests and ARP Reply messages to establish paths. Note that only ARP frames (or special broadcast frames in failure cases) discover or create new paths.

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. This mechanism ensures loop-free flooding of frames over all links. *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).

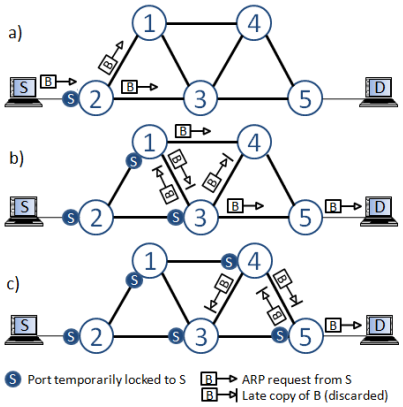

Figure 1: ARP-Path discovery from host *S* to host *D*. The small bubbles on the links show which bridge port locked *S*'s address

2) 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.

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, but do not produce address learning.

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.

B. Advantages

ARP Path protocol operates on an on-demand basis, thus it has several advantages over other protocols that build routes proactively like layer two link state routing protocols [2][3].

- *Minimum Latency*.
- *Zero configuration*.
- *Simplicity*.
- *Load distribution and path diversity*.
- *Scalability*.

III. ARP PATH NETWORK DEMONSTRATION

The demo network is shown at Fig. 2. The objective of the demonstration is to show the ARP Path bridges protocol operating in a network of ten ARP PATH bridges with different hardware and protocol implementations, mostly Openflow [5] based plus two OpenWRT based ARP-Path switches running a Linux-based ARP Path implementation. Although ARP Path is a fully distributed protocol, Openflow enables a simple implementation and validation of the protocol with commercial hardware.

Demo shows performance, reconfiguration speed and robustness of the ARP-Path bridges concept in small and medium network without the spanning tree protocol or any ancillary routing protocol at layer two. We demonstrate the reconfiguration, compatibility with hosts and internet connections and the absence of broadcast storms or other infinite loops. Demo also shows the flexibility and power of Openflow to implement new protocols in real networks.

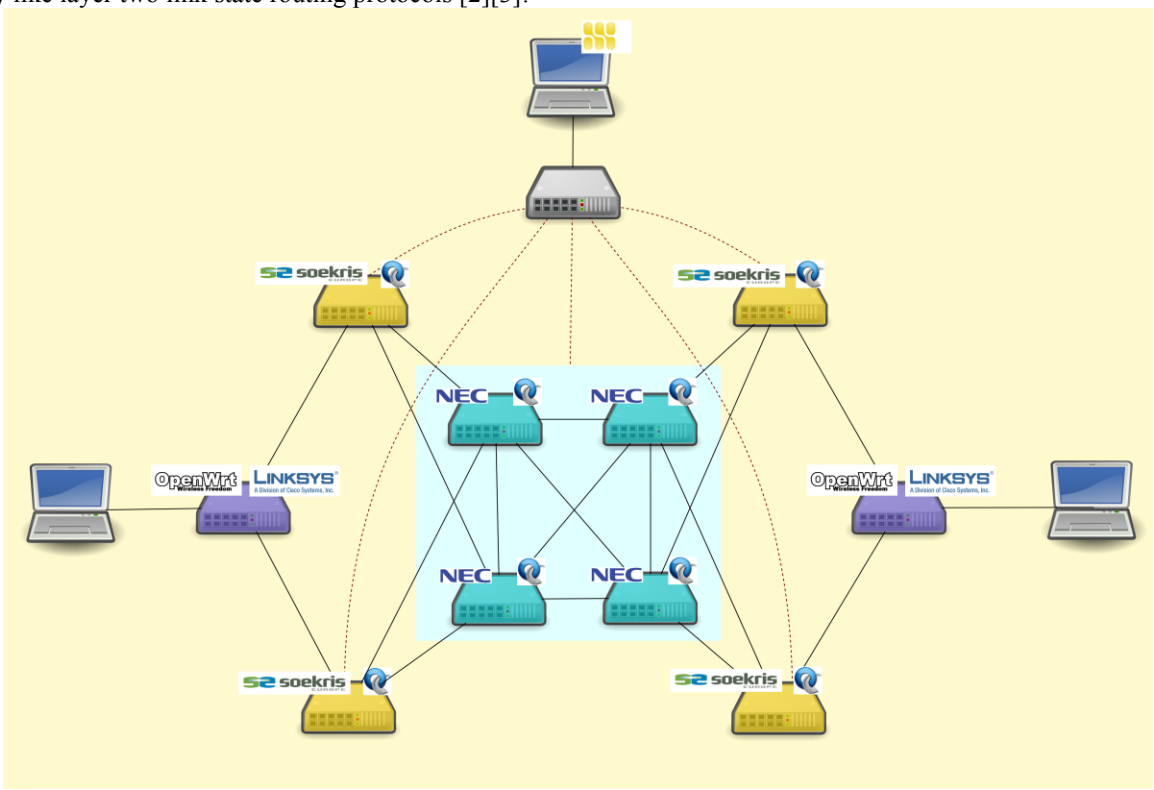

Figure 2. Demo network of ARP Path switches combining Openflow (NEC and Soekris) switches and OpenWRT implementations

A. Network elements

The network of Openflow switches is as follows: a core NEC switch (model P8800 or NEC Programmable Flow PF5240) that is configured as four independent Openflow capable switches, and four Soekris net5501 boards executing Openflow software. All Openflow switches are controlled via a NOX controller that handles the flow processing and implements the logic of the ARP Path protocol of the bridges.

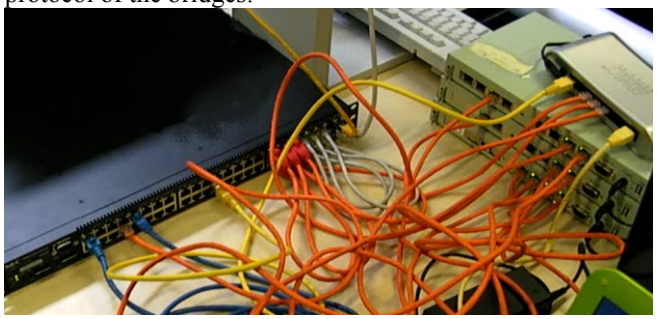

Figure 3. NEC switch and Soekris boards

Regarding the two OpenWRT devices, ARP-Path bridge protocol was first implemented on a Linux 2.6 kernel and operates in kernel and user space using *ebtables*. This implementation has recently been ported to OpenWRT, a Linux distribution for embedded devices [6] to obtain cheap ARP Path bridges (40 € to 20€ /node) that makes possible to build big pilot networks. In the demo network, each OpenWRT is connected to two Soekris boards, acting as access switch, as shown in Fig. 4.

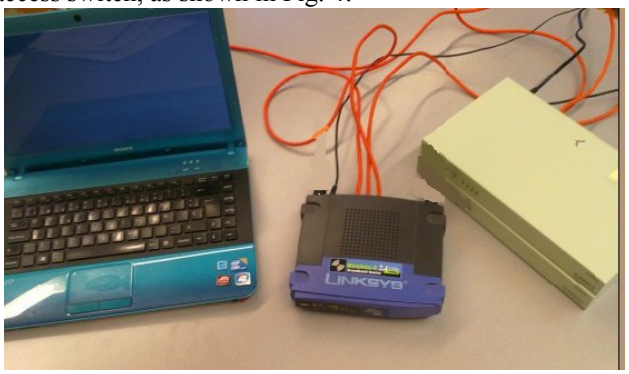

Figure 4. Side of network (OpenWRT and Soekris)

B. Demo description

First, pings are sent to test connectivity between hosts and round trip time measurements are performed, showing the differences between path set up (processing ARP Requests requires access to the NOX controller and increases the delay and normal switching (addresses are already learnt) delay of fraction of millisecond.

Second, hosts are configured as multicast video servers with several Videolan server screens active and are also connected as clients and to video web pages. All network *links are connected and active at the same time and the shortest (direct in this case) paths are selected. Both video streaming with UDP and or HTTP are used. One of the host is, while receiving a video, disconnected and reconnected at other switch port in the network. Video streaming is resumed normally. Links between hosts are removed sequentially and the video reception shows that the path is repaired after each link is unplugged. Whaen all redundant links are extracted, video reception stops. When a link is plugged again, video reception resumes.

Path set up time with Openflow, measured with repetitive pings between source and destination and removing and restoring Ethernet cable is aprox. 230 msec (MAC is not in table at ARP-Path switches). Once the path is established, it takes only 0.6 msec for a ping to go and get back. Path recovery time after link failure is approximately 47 msec. Openflow controller processing introduces an additional delay of approximately 10 msec from "packet in" request to the flow-mod response.

C. Commands

Basic commands for Openflow execution are as follows:
-To associate Ethernet ports to Open flow and identify the device for the controller (x = number) and list the ports
~/openflow/udatapath/ofdatapath --detach punix:/var/run/dp0 -d 00000000000x -i <ports>
-Then execute Openflow (IP of the Open flow NOX controller):
~/secchan/ofprotocol unix:/var/run/dp0 tcp:<controller ip>
-In the controller PC, NOX is started with module *modswitch*, the module that implements ARP Path protocol logic:
~/nox/build/src/nox_core -v -i ptcp:modswitch

D. Equipment at conference venue

The required equipment (provided by us) is listed below:

- Three laptops and AC/DC power supplies
- AC power for laptops
- Optional: wired access to internet

Demo requirements are: Internet access (preferably wired), AC power, Space required: table 60 cm x 120 cm, set up time is 40 minutes.

E. References

[1] IEEE 802.1D-2004
[2] M. Seaman. Shortest Path Bridging. http://www.ieee802.org/1/files/public/docs2005/new-seaman-shortestpath-par-0405-02.htm.
[3] Transparent interconnection of lots of links (TRILL) WG. Available on line at: http://www.ietf.org/html. charters/trill-charter.html
[4] Ibanez G. et al. ARP-Path: ARP-Based, Shortest Path Bridges. *IEEE Communication Letters July 2011*.
[5] Openflow: http://www.openflow.org
[6] OpenWRT. https://openwrt.org
[7] G. Ibáñez et al. A Simple, Zero-configuration, Low Latency, Bridging Protocol".LCN 2011 demos. www.ieeelcn.org/prior/LCN35/ lcn35demos/lcn-demo2010_ibanez.pdf
[8] G. Ibanez et al. Implementing ARP Path low latency bridges on NetFPGA. *Sigcomm demos 2011*, http://conferences.sigcomm. org/sigcomm/2011/papers/sigcomm/p444.pdf