ARP-Path: ARP-based, Shortest Path Bridges

Guillermo Ibáñez, Member, IEEE, Juan A. Carral, José M. Arco, Diego Rivera, Aarón Montalvo

***Abstract*— This letter is a summary proposal for an evolution of the Ethernet transparent bridge paradigm that provides simple, shortest path bridging in campus networks. ARP-Path Ethernet Switches set up an on-demand path between two hosts just reusing and flooding the standard ARP request frame through all links and confirming the path reaching to the destination host with the ARP reply frame. ARP-Path uses the standard Ethernet frame format, is fully transparent to hosts and does not require spanning tree or link state protocol. Simulation results show superior performance to spanning tree and similar to shortest path routing, with lower complexity. Our implementations confirm backward compatibility, robustness and performance.**

***Index Terms*—Ethernet, Routing bridges, Spanning Tree**

I. INTRODUCTION

Ethernet switched networks offer important advantages in terms of price/performance ratio, compatibility and zero configuration, but the spanning tree protocol (STP) [1] limits the performance and size of Ethernet networks. Current proposals under standardization, 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. However, they have significant complexity both in terms of computation and control message exchange and need additional loop control mechanisms. We present ARP-Path Ethernet Switching (ARP-Path, for short), a simple, zero-configuration protocol for metro, campus, enterprise, and data center networks that enable the use of all available links without link state routing.

II. ARP-PATH PROTOCOL

An ARP Path is the fastest (and unique) path created by an ARP Request frame that reaches its destination host and is confirmed backwards by the corresponding ARP reply.

A. *ARP-Path Set up*

The *path discovery* process is described in fig.1 and works as follows. Host S sends a standard ARP Request on an Ethernet broadcast frame B to resolve the IP address of a given host D. The ingress bridge 2 receives the frame from S and temporarily associates (*locks*) the global MAC address of S to its arriving port and blocks the learning of S address on all other ports (i.e., further broadcast frames from S, arriving to other input ports of bridge 2 will be discarded as late frames from S). Then, it broadcasts B in the standard way (fig.1 a). Bridges 3 and 1 behave as bridge 2, locking S address to the arriving port of B and also broadcasting B through all other ports but the arriving one. Hence, duplicate copies of B would arrive to bridges 3 and 1 sent by each other but they arrive at a different port from the one already locked to S, so they will be discarded (fig.1 b). Then, bridges 4 and 5 will process B in the same way. At last, a copy of B will arrive to the destination host D. A chain of bridges with an input port locked to S (i.e. a temporary path) is now active between S and D (fig.1 c).

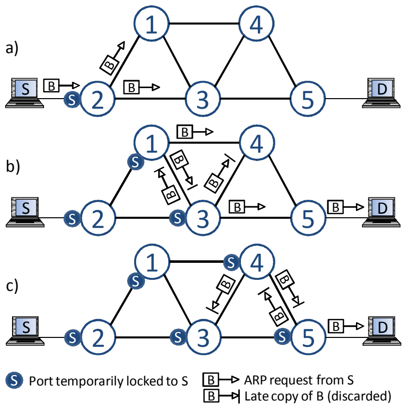


Figure 1. ARP-Path discovery from host S to host D.

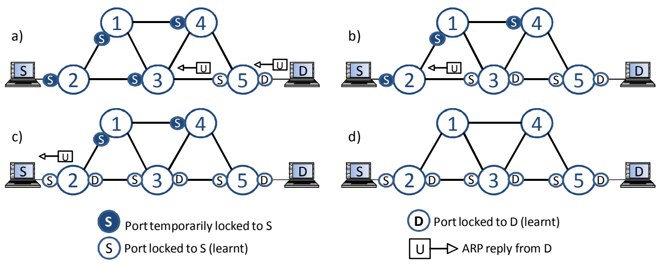
The *path discovery* process creates a transient broadcasting tree of locks to S address (a tree rooted at the bridge serving host S that reach to every other network bridge). This tree blocks the "learning" of S address but does not block the forwarding of other ARP frames coming from S; they are accepted if they arrive to the "chosen" port and discarded otherwise. A copy of every ARP issued by S is guaranteed to arrive to the port chosen by the first ARP (although, for successive ARPs, it might be not the first one received). In fact, the first ARP request sets up the tree, while subsequent ARP requests from S "follow the trail" (the tree of locks) as long as it is in place.

Fig. 2 shows the process of *path confirmation* from D to S. Host D sends the ARP Reply towards host S on a standard unicast U frame. Bridge 5 learns D location (port) in the standard way and confirms the S lock (S address is also learned). Thus, bridge 5 has now confirmed routes to S and D. Standard cache timers are activated for both entries. Then it forwards U via the port previously locked to S (see fig. 2 a)). Bridge 3, and then Bridge 2, will process its own copy of U, learn D location, confirms S lock and forward U through the port associated to S. Hence a copy of U would reach host S (fig. 2 b) and c)). Eventually, the locks established in bridges 1

Manuscript received April, 2010. 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).

G. Ibañez, J. A. Carral, J. M. Arco, D. Rivera and A. Montalvo are with Departamento de Automatica, University of Alcala, Spain. (phone: 34-918856927; fax: 34-918856924; e-mail: guillermo.ibanez@uah.es)

and 4, which do not belong to the confirmed path, would expire (fig. 2 d).


a)
b)
c)
d)
1
2
3
4
5
S
D
U
Port temporarily locked to S
Port locked to S (learnt)
Port locked to D (learnt)
ARP reply from D
Figure 2. ARP-Path confirmation from host D to host S.

B. *ARP-Path Restoration*

An established *ARP-Path* (i.e. a chain of learned addresses at bridge ports) may get broken at some point either by the expiration of an address timer or by a link failure. The failure of a link connecting two ARP-Path bridges provokes the flushing of all MAC addresses associated to the two ports of that link. The same happens at all ports of a node in case of node reboot.

Whenever a bridge receives a frame with an unknown destination address (i.e. the address is not associated to any port), a new path must be rebuilt from the source bridge (i.e. the bridge serving the source host). A bridge receiving an unknown destination unicast frame would return a *path_fail* message towards the source host. This message is addressed to the *all_ARP-Path* MAC multicast group and carries as payload the header (source and destination MAC addresses) of the rejected unicast frame. Frames sent to this multicast group are (only) processed by ARP-Path bridges.

Each bridge in the path would relay back the *path_fail* message, based on the source MAC carried in the payload, until it reaches the edge bridge serving the host source of the rejected unicast frame. This bridge is responsible for starting a new path discovery process to replace the broken one.

The source edge bridge sends a *path_request* message addressed to the *all_ARP-Path* multicast group. This address is used so that all ARP-Path bridges receiving the frame know that they must process the frame before forwarding it. The *path_request* message carries both the source and destination host MAC addresses as payload (extracted from the *path_fail* message) and is processed as an ARP Request message by all ARP-Path bridges (i.e. it works as a path discovery process). The bridge serving the destination host will reply back with a *path_reply* message which follows back the trail of locks set by the *path_request*, thus confirming the new path (i.e. it works as a path confirmation process, but between edge bridges).

C. *Frame processing*

A frame *F* (with destination address *DA*, source address *SA* and carrying a data field '*Data*') received at port *P* of an ARP-Path switch, is processed as shown in Fig. 3.

According to protocol rules, a given address *A* may be unknown (*UNKNOWN*) to a switch, may be known and associated to port P_A (*LEARNT*(P_A)) or may be temporarily associated to port P_A (*LOCKED*(P_A)). The values (DAf, SAf) encoded within a *path_fail* message and (DAr, SAr) encoded within a *path_request* message respectively denote the destination and source MAC addresses of the failed and requested paths.

```
If ('DA' is Bcast or Mcast (DA ≠ ALL_ARP-PATH))
    If ('SA' is UNKNOWN) {Lock 'SA' to port 'P'; Broadcast F}        [1]
    Else If (P = $P_{SA}$) {Broadcast F; Refresh entry}
        Else {Discard F}
If ('DA' is Mcast (DA = ALL_ARP-PATH))
    If ('Data' contains 'path_fail message ([DAf, SAf])')
        If (SAf is a Bridge leaf) {Send a 'path _request ([DAf, SAf])'}
        Else {Forward F back to SAf}
    If ('Data' contains 'path_request message ([DAr, SAr])')
        If (DAr is a Bridge leaf)
            {Send a 'path _reply ([DAr, SAr])' back to SAr}
        Else {"Do as in [1] using SAr as source address"}
    If ('Data' contains 'path_reply ([DAr, SAr])')
        {"Do as in [2] using SA=DAr, DA=SAr, 'Data' is ARP_Reply"}
If ('DA' is Ucast)
    If ('SA' is UNKNOWN)
        If ('DA' is UNKNOWN)
            {Send a 'path_fail message ([DA, SA])' back to SA}
        Else
            If ('Data' contains a 'ARP_Reply')
                Learn SA at port P
                If 'DA' is LOCKED ($P_{DA}$) {Learn DA at port $P_{DA}$}
            Forward F via port $P_{DA}$
    Else /* 'SA' is LOCKED ($P_{DA}$) or LEARNT ($P_{DA}$) */        [2]
        If (P = $P_{SA}$)
            Learn SA at port P or refresh SA entry
            If ('DA' is UNKNOWN) {Send a 'path_fail ([DA, SA])' to SA}
            Else If ('DA' is LOCKED ($P_{DA}$)) and
                    ('Data' contains a 'ARP_Reply')
                {Learn DA at port $P_{DA}$; Forward F via port $P_{DA}$}
        Else If ('DA' is UNKNOWN) {Send a 'path_fail ([DA, SA])' to SA}
            Else {Forward F via port $P_{DA}$}
```

Figure 3. ARP-Path protocol frame processing pseudo-code.

III. Evaluation

We compare ARP-Path protocol with standard bridges (STP/RSTP) and shortest path bridges (SPB) regarding computational complexity, message overhead and results from software simulations.

A. *Stored state and complexity*

The main advantage of the ARP-Path protocol is its simplicity. ARP-Path bridges basically operate as standard bridges with some increase in stored information but without neither link state nor spanning tree protocol computations.

1) *Stored information*

ARP-Path bridges cache, in stationary state (i.e. once the address is confirmed (learned)), the same information than standard bridges (MAC address to bridge port associations). Additionally, they require some extra information, but small, stored at every bridge to implement to locking mechanism: store locked addresses (one per bridge and per path) during the ARP based path discovery and confirmation process. This locking mechanism can be seen as a separate function for loop free broadcasting (flooding) and path/address learning.

2) *Computational complexity*

ARP-Path bridges do not need to make any route computation at all. On the other hand, SPB implements a link state protocol (IS-IS based) to acquire the network topology, and apply the Dijkstra algorithm to compute the shortest path routes. Its computational complexity is, for a network of N bridges, $\theta(N^2)$, with a minimum of $N{\cdot}logN$ (using heap based implementations).

Additionally, ISIS SPB must guarantee path congruency to keep the backward learning mechanism safe (prevent oscillations). Thus, every SPB bridge must compute the shortest path trees of every other bridge to assure both bridges select the same path when several equal cost paths are available. Hence, the computational complexity is increased by a factor of N, resulting in $N^2{\cdot}logN$, that may compromise scalability and reconfiguration times in big networks [4].

B. *Message overhead*

ARP-Path bridges do not periodically exchange routing information (like IS-IS LSAs). Instead, the standard ARP message exchange is reused to set up paths when needed.

1) *Overhead due to the broadcasting (loop free) mechanism*

ARP-Path slightly increases the number of broadcast frames when compared to tree based broadcasting. ARP-Path bridges broadcast frames over all inter switch links, instead of only via spanning tree links. In a network of N bridges, interconnected by L links and serving H single connected hosts, STP produces $C=H+N-1$ copies of every broadcast addressed frame (to reach every host) while ARP-Path needs $C=2{\cdot}L-(N-1)+H$ [5]. Hence, ARP-Path produces $2{\cdot}(L-N+1)$ more copies (twice the number of redundant inter bridge links).

For instance, in a data center network like the one shown in fig. 4, but extended to serve H=4000 hosts (i.e. 4000 host links), with 40 access bridges, 4 core bridges (N=44) and L=86 inter-bridge links the total number of copies of a broadcasted frame would be: C=4043 using STP and C=4129 using ARP-Path (only 2.13% higher).

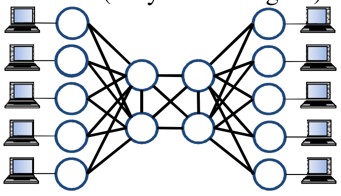


Figure 4. Typical two level data center network.

It is well established that ARP broadcast traffic limits scalability of layer two networks because it poses a significant load to all network hosts that must process all ARPs received. Recent proposals aiming to broadcast minimization using ARP proxies, like Etherproxy [6], are well suited for implementation inside ARP-Path bridges with low additional complexity (basically adding IP addresses to bridge caches). ARP Requests are intercepted and replied at the first edge bridge by the proxy.

Ref. [6] provides a detailed performance evaluation of ARP proxy. In summary, hit rates vary with traffic distribution profiles in the range 60-80% and may reach 100% with proactive refreshing of close to expiration address entries (at the cost of longer cache tables). Only those ARP Requests that miss at proxy cache will get through the first edge bridge. Hence, the number of ARP Requests received at a host is reduced accordingly and so is the number of copies throughout the network.

In our example, using the proxy strategy together with ARP-Path would reduce the number of broadcast copies to a 20% (from 4129 to 825.6), assuming a modest 80% hit rate.

C. *Measurement results*

All three protocols were simulated in Omnet++ on the 16 node pan-European core reference mesh network [7]. UDP traffic flows are exchanged between arbitrary pairs of nodes (8 flows per run); flow rates increase from 100 kbps up to 6 Mbps. Link delays range between 1and 3 ms.

Fig. 5 shows the average percent link load obtained at the most loaded link versus the percent load at a client host link (all links are 100 Mbit/s). ARP-Path produces similar results to Shortest Path Routers while STP performance is the worst case by far.

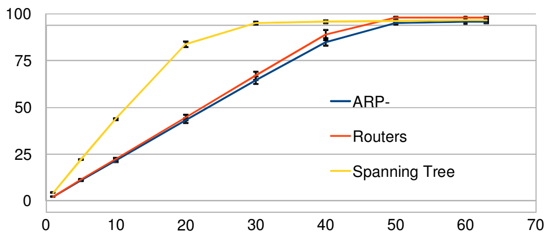


Figure 5. Throughput comparison in panEuropean network.

Fig. 6 shows the average delays obtained with increasing traffic. Again, ARP-Path delay with low load is similar to shortest path routers and lower with increased loads while STP always produces longer delays.

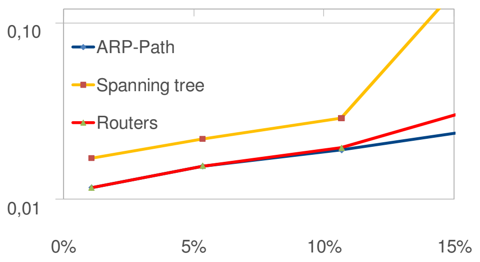


Figure 6. Delay (seconds) of reference network vs. percentage of load.

ARP-Path has been successfully implemented in two platforms: Linux and Openflow/NetFPGA boards and demonstrated at the last LCN congress [5][8].

References

[1] IEEE 802.1D-2004 IEEE standard for local and metropolitan area networks-Media access control (MAC) Bridges.
[2] Shortest Path Bridging. http://www.ieee802.org/1/pages/ 802.1aq.html
[3] Transparent interconnection of lots of links (TRILL) WG. http://www.ietf.org/html.charters/trill- charter.html.
[4] J. Farkas, Z. Arató. Performance analysis of shortest path bridging control protocols. GLOBECOM 2009 Honolulu pp. 4191-4196.
[5] G. Ibanez et al. “FastPath Ethernet Switching: On-demand Efficient Transparent Bridges for Data Center and Campus Networks“. LANMAN May 2010.
[6] K. Elmeleegy, A. L. Cox, “Etherproxy: Scaling Ethernet by suppressing broadcast traffic,” in Proc. IEEE INFOCOM, Rio de Janeiro, Apr. 2009
[7] NRS Ref. Net. www.ibcn.intec.ugent.be /INTERNAL/NRS/index.html
[8] G. Ibáñez et al. A Simple, Zero Configuration, Low Latency Bridging Protocol. LCN 2010 demos. http://www.ieeelcn.org/prior/LCN35/lcn35 demos/ lcn-demo2010_ibanez.pdf