This document is published in:

IEEE Local Computer Networks (LCN). 37th Conference on (2012) pp. 244-247

DOI: 0.1109/LCN.2012.6423619

Flow-Path: An *AllPath* Flow-based Protocol

Elisa Rojas, Guillermo Ibáñez, Diego Rivera, Juan A. Carral
Depto. Automática. Universidad de Alcalá (UAH)
elisa.rojas.sanchez@gmail.com, guillermo.ibanez@uah.es, diego.rivera.pinto@gmail.com, jac@aut.uah.es
Alcalá de Henares, Spain

***Abstract*— This paper describes Flow-Path, an *AllPath* flow-based switching protocol that features improved load adaptive properties. Upon arrival of a new flow to the network, it explores every possible path reaching from source to destination host and selects the lowest latency path at the moment. It is based on the same basic principle than ARP-Path, that is, snooping the ARP protocol dialog (request and reply messages) to explore all available paths at the same time that address resolution takes place, but it is flow-based instead of source address-based. While preserving the main advantages of ARP-Path: shortest path bridging exploiting the full network topology, Flow-Path has the advantages of full independence of flows at the time of path creation and guarantees path symmetry (congruency) and increased path diversity. Flow-Path thus improves load distribution, at the expense of increased address table size in each bridge.**

Keywords: Ethernet; Routing bridges; Shortest path bridges

I. INTRODUCTION

Ethernet switched networks offer important performance and cost advantages for local, campus and metropolitan networks, i.e., an excellent price/performance ratio, with a high compatibility between elements, and a simpler configuration than IP. But the Spanning Tree Protocol (STP/RSTP) [1] severely limits the network size and performance by blocking all redundant links to prevent loops, thus limiting infrastructure utilization and increasing latency. In this situation, simple, zero configuration protocols that remove the limitations of RSTP and, at the same time, allow scaling Ethernet to deal with bigger single IP subnets, might become the key to boost its deployment on campus, data center and even enterprise networks.

ARP-Path protocol has recently emerged as a shortest path -more precisely, low latency path- bridging alternative [2-3]. It is based on evolved bridging mechanisms that take advantage of the information conveyed by the ARP protocol message dialog to construct the forwarding table. The ARP protocol *request* and *reply* messages are snooped at ARP-Path bridges to select low latency paths with zero configuration and full transparency to both hosts and switches.

The high degree of path diversity (load sensitive path selection) achieved by the ARP-Path protocol produces excellent results both in terms of latency and overall throughput [2-3], but there are some special scenarios where overall performance may degrade. In networks with very asymmetric traffic patterns (for instance, nearby network servers) the paths selected may not be optimal. Also, there are some situations where path *congruency* (full coincidence of paths in both traffic directions) may not be guaranteed, as explained later in the paper. Congruency is not a mandatory requirement for protocols like ARP-Path which do not learn host locations from unicast data frames, but it may simplify connectivity fault management and may be an interesting feature in some network scenarios.

In this paper, we present Flow-Path, a flow-based protocol inspired in the same operation principles than ARP-Path, which sets up a specific path for each data flow (identified by the pair of hosts involved). Both ARP-Path and Flow-Path protocols belong to a new category of bridges that we name *AllPath*, because all paths of the network are simultaneously explored with a broadcast frame forwarded over all network links to find a path or set a multicast tree.

We have tested the protocol by using a simulator written on OMNeT++ [4]. Also, it has been implemented by using Mininet [5], Soekris[6] boards and the OpenFlow [7] standard. Details on the implementation results are not given due to lack of space.

II. PROTOCOL DESCRIPTION

Flow-Path relies on the same basic principle that ARP-Path: the race between replicas of a flooded ARP Request to discover the fastest path. Therefore, we take the ARP-Path behavior as a reference in the description.

However, ARP-Path protocol may produce transitory asymmetric paths in each direction under certain conditions. Let's consider the situation shown in Fig. 1 where an active flow between A and C is in place while a new flow between B and C is started by B. When the request sent by B arrives to switch 3, it is broadcasted to switches 5 and 6 and then by switch 5 via link 5-6. It may happen that, if link 3-6 is heavily loaded, the request coming to switch 6 via link 5-6 arrives to switch 6 before the copy sent via link 3-6. Thus, the reply sent by C in response will follow back the path 6-5-3. Now, switch 3 has a port confirmed to reach C (via link 3-6) and a control message saying C must be reached via link 3-5. Switch 3 may choose to keep the old entry so a partially asymmetric path between B and C is constructed (path 2-3-6 from B to C and 6-5-3-2 from C to B) or may switch C entry to link 3-5 moving active flows from one link to another and, again, creating a partially asymmetric path (now between A and C).

Furthermore, the load distribution capabilities of the protocol will be reduced especially if the destination host C is a big attractor of flows (i.e. a server).

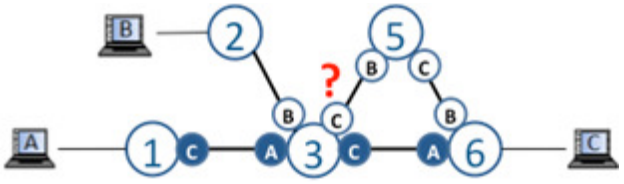

Figure 1. Asymmetric path (between B & C) formation.

In order to provide a better load distribution with per-flow granularity and to avoid the above-mentioned situations, we developed the Flow-Path protocol. It creates fully bidirectional paths by associating ports to flows instead of to single source addresses.

A. *Path set-up*

Consider host A (see Fig. 2) wants to communicate with host C by using the IPv4 protocol (for IPv6 the mechanism is similar but using Neighbor Discovery protocol instead). First, A will flood an ARP Request to obtain C's MAC address. According to the ARP-Path operation rules, every switch receiving the request will lock (associate) the first-arrival port, however, unlike the classical ARP-Path, the port is associated to the flow from A to C instead of only to A's MAC address. This is done by inserting the source and destination addresses in the *Locked Table* (LT). Then, the ARP Request is forwarded via all other ports but the receiving one. Later copies of the request, received at other ports of the switch, are simply discarded, just by checking that their entry is associated to another port. Since the ARP Request does not carry C's MAC address (yet unknown), the protocol relies on the IP addresses of A and C to fully identify the flow until C's MAC address is later learnt. Table I shows the *Locked Table* once the request has been processed at a switch.

TABLE I. LOCKED TABLE ENTRY IN A FLOW-PATH SWITCH

Key (Src-Dst)	Src IP	Dst IP	Port	Timer	State
A-FF:…:FF	IP_A	IP_C	1	0	Locked

When the fastest arriving copy of the request is finally delivered to host C by switch 6, it issues the corresponding ARP Reply. As shown in Fig. 2, the ARP Reply from C is used to confirm the path, i.e. to create the *Forwarding Table* (FT), in the reverse direction. Every switch in the path of the ARP Reply will learn C's MAC address and confirm a pair of flow's entries, one for each flow direction, in its forwarding table. Then, the reply is forwarded to host A via the port previously learnt in the *Locked Table*. This mechanism ensures a symmetric path is always constructed. After the ARP Reply, the *Forwarding Table* of a switch will look as shown in Table II. Note that IP addresses are no longer needed to identify the flow; frame forwarding is performed based on the tuple <source, destination> MAC addresses.

TABLE II. FORWARDING TABLE ENTRY IN A FLOW-PATH SWITCH

Key (Src-Dst)	Port	Timer	State
A-C	1	0	Confirmed
C-A	2	0	Confirmed

In the unlikely event that two hosts (A and C) send an ARP Request to each other at the same time, different paths may be selected for each direction, but the switches would detect it and only confirm one of them by simply applying a priority rule based on A and C MAC addresses.

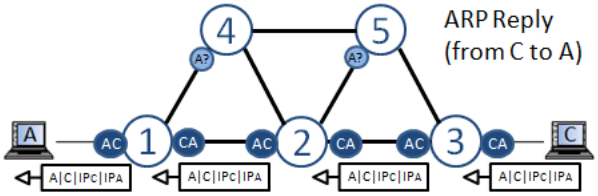

Figure 2. Flow-Path path confirmation

Both ARP-Path and Flow-Path rely on the same basic principles: *i)* the *locking mechanism* to guarantee loop-free forwarding and, *ii)* snooping the information conveyed by the ARP request and reply frames to set-up on demand paths (the fastest available one at the time). But the paths constructed differ in that they are flow-oriented instead of host-oriented in Flow-Path's case. Hence, a different path may be set-up for each pair of hosts thus fully exploiting the available resources on the network. This may be important around network hot spots (i.e. network servers that concentrate many flows) to balance the traffic on the available paths at the cost of increasing the forwarding table sizes. The Flow-Path protocol also guarantees path symmetry in both directions (path *congruency*).

B. *Path Recovery*

If a link or switch fails, the ports connected to that failure point will detect it and will delete all the entries associated to this port. When an unicast frame arrives to this switch it will find no route to destination and the path recovery mechanism will start for that flow. Note that, opposite to spanning tree protocols, only the addresses related with the failing port are flushed and path recovery is performed only for the active flows and when it is needed.

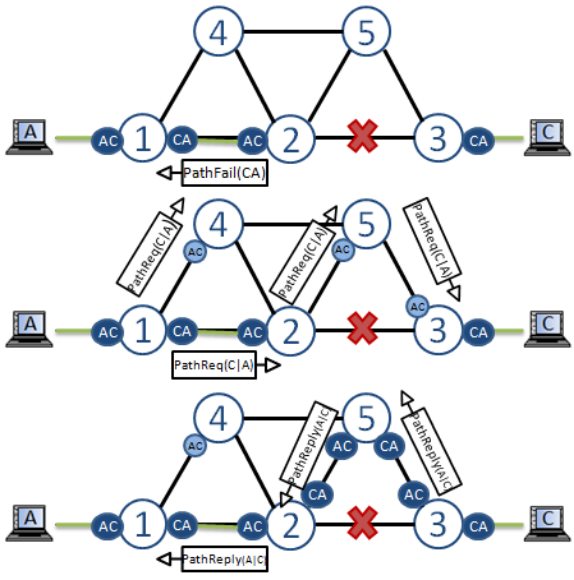

Figure 3. Example of path recovery with looped back frame.

The new and simplified path recovery mechanism (see Fig. 3) performs a loopback of the unknown unicast frames towards the source edge bridge, instead of generating a broadcast *Path_Fail* message, thus avoiding broadcast and special frame processing at this critical bridge. A frame is considered a *looped back* frame at a bridge when it is received via the port associated to its source-destination address pair in the opposite direction. Looped back frames are forwarded with the same forwarding logic, but reverted (i.e. frame source address is used instead of destination address) until they reach the each bridge. At every bridge the received looped back frame flushes and sets the flow under recovery status for a short time, to prevent consecutive frames to trigger redundant path recovery processes. When the looped back frame reaches the source edge bridge, the bridge detects its source address as a directly connected host and sends a new ARP packet to set up a new path.

III. EVALUATION

The evaluation of the Flow-Path protocol is mainly focused on the differences with ARP-Path. As we already pointed out, Flow-Path offers some interesting advantages such as an even better load balancing and guaranteed path symmetry, at the cost of a slightly more complex pseudocode, but still simple, and an increase of table sizes (state stored). In the following paragraphs, we will analyze this advantages and disadvantages theoretically and also with the Omnet++ simulator.

A. *Theoretical Analysis*

As it is already known, a key difference is the size of the *Forwarding Table* (FT), because those entries in Flow-Path are created per flow (source and destination pair) instead of per source address as in ARP-Path. Since the number of flows is a power of the number of hosts, this would suggest the Flow-Path *Forwarding Tables* would have much more entries. However, this is not completely accurate.

In Flow-Path, every flow creates two FT entries (one to forward to destination and the other to forward back to source, or viceversa) at every bridge. Therefore the number of total table entries in all the switches in a topology is twice the number of flows multiplied by the average number of switches traversed by a flow path:

$$T_{FP} = F * 2 * b \qquad (1)$$

Where T_{FP} is the total number of table FT entries in the topology, b is the average number of switches per path and F is the average number of active flows in a network for any instant of time.

For the ARP-Path protocol, entries are not created based on flows but individual addresses, i.e. every host will create one entry (to forward to that host address) at every bridge. Therefore the expression would be:

$$T_{AP} = H * (b + L_e) \qquad (2)$$

Where T_{AP} is the total number of FT entries in the topology, H the average number of active hosts in the network for any instant of time, and L_e the number of *extra links* added when an entry is shared by different flows. Note that, in the worst case, $b + L_e = S$, where S is the total number of switches in the topology.

By dividing both expressions, (3) is obtained:

$$R = \frac{F*2*b}{H*(b+L_e)} \leq \frac{\frac{H*(H-1)}{2}*2*b}{H*S} = \frac{(H-1)*b}{S} \qquad (3)$$

As it can be seen, the maximum ratio for the number of table entries is not as bad as expected, since it is true that it depends on the size of the topology, i.e. H, but it also depends on the topology shape, i.e. b/S, which is a factor always lower than 1 and reduces R.

One way to estimate the number of entries at the forwarding tables makes use of the number of active flows measured in data centers (likely higher than for campus networks). In [8] the number of measured active flows at each machine is lower than 10 flows during 50% of the time, between 80 and 100 flows for less than 5% and is never greater than 100 flows. Taking into account the distribution curve we use an average of 10 active flows per host and calculate the number of FT entries per switch:

$$T_{FP\ per\ switch} = F * 2 * \frac{b}{B} = 20 * H * \frac{b}{B} \qquad (4)$$

For a data center network topology with 2500 hosts, 50 hosts per edge bridge, 50 edge bridges and 4 core bridges, i.e. average core bridges traversed equal to 2, we obtain 25000 (20*2500*2/4) entries per core bridge of a theoretical total of 6247500 (2500*2499) table entries.

B. *OMNeT++ Simulation Analysis*

The protocol has been tested by using the OMNeT++ INET simulation environment. Flow-Path operation has been proven, both path creation and recovery were successfully tested in different topologies. Paths were created avoiding loops and path congruency was always guaranteed.

In order to validate the analysis of the previous section, the sizes of the *Forwarding Tables* at each node were studied and compared for both protocols (ARP-Path and Flow-Path) and they grew as expected when the number of flows was increased.

However, one of the most remarkable advantages of the Flow-Path proposal over ARP-Path protocol is being capable of improving the load distribution in the network even more in certain circumstances. When two or more flows share the same destination address, ARP-Path will create only one entry per switch to reach that destination, while in Flow-Path, it is possible to have more than one entry for a specific destination address because it might belong to different flows and Flow-Path distinguish those flows in order to build the paths. For this reason, if a specific destination needs to handle a lot of traffic, ARP-Path will have only one path to reach it, while Flow-Path will be able to create one path per flow, separately, so that the load distribution is better. Notice that ARP-Path is really good at load balancing in a global vision for a whole topology, while Flow-Path is not only at that, but also for specific scenarios

when a concrete host needs to handle much traffic, at the cost of a higher number of FT entries.

A comparison was performed with a square mesh topology with 9 switches and 6 groups (from A to F) of 25 hosts connected at each side of every switch row, as shown in Fig. 4. The group of hosts A from the upper left switch has been configured to send UDP traffic to hosts located in the bottom right group F.

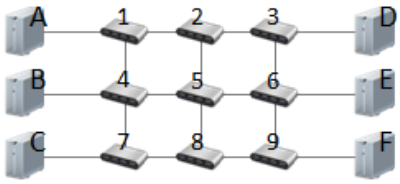

Figure 4. Topology used for load distribution simulations.

In order to show the worst and best case for Flow-Path against ARP-Path, two different simulation configurations were prepared. For each simulation, we have executed sixteen repetitions, with different random seeds for traffic configuration and configured a total number of 25 flows. In each case, the channel utilization percentage was measured and compared for each link in the network to show the load distribution.

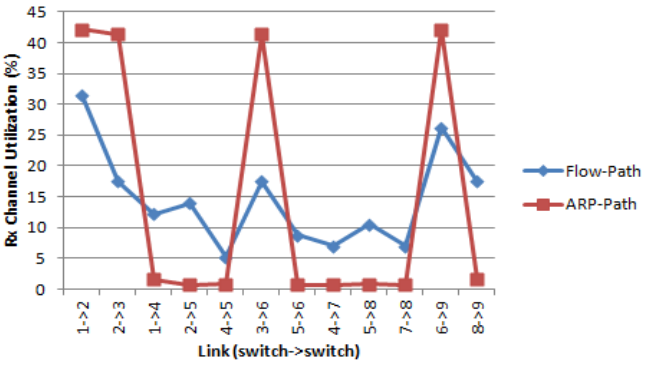

Figure 5. Load distribution comparison between Flow-Path and ARP-Path (best case for Flow-Path)

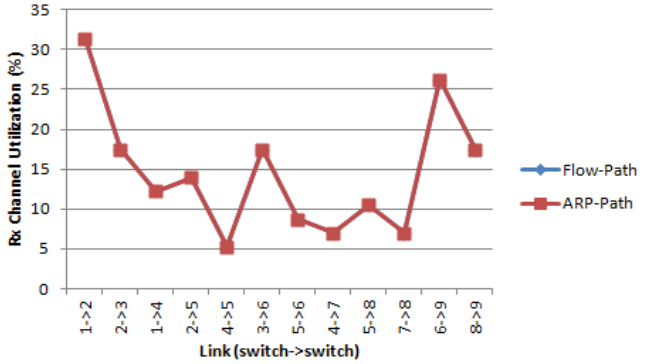

Figure 6. Load distribution comparison between Flow-Path and ARP-Path (worst case for Flow-Path)

To simulate the best case for Flow-Path, we configured the 25 hosts from A to send traffic to just one host of F, so that ARP-Path would only create one shared path to that destination, while Flow-Path would create 25 paths, equal to the number of flows, allowing load distribution. The worst case was simulated by creating independent flows from the 25 hosts of A to the 25 hosts of F, so that the load distribution would be almost exactly the same for Flow-Path and ARP-Path, since both would create independent paths for every flow.

The best case for Flow-Path is shown in Fig. 5, where ARP-Path concentrates all the traffic for the destination in one single path (1-2-3-6-9) while Flow-Path creates 25 different paths, distributing the traffic and avoiding traffic peaks. The worst case for Flow-Path is shown in Fig. 6, where the load distribution is exactly the same for both protocols, as expected (since the seed for both simulations is the same). Note that there is still load distribution in the network, but it is considered the worst case for Flow-Path because there are no differences with ARP-Path, although load distribution in the network is good anyway.

With these results, the use of Flow-path for some topologies, like data centers in which one server may be communicating with many hosts at the same time, might be advisable to improve load distribution. This is because with the original ARP-Path protocol many paths from hosts to this server may have coincident links, reducing its inherent load distribution. Paths with Flow-Path are independently found per source-destination pair, since each path is created in a unique way for every flow (between the server and each host) and the path with lowest latency at that moment will be selected, so adapting to the current traffic.

IV. CONCLUSION

The main advantages of the Flow-Path protocol are the independence among flows at the time of constructing their paths, providing even better load distribution than ARP-Path in some scenarios (like hot spots), and the guaranteed path symmetry, at the cost of an increased table size and some increment in processing.

ACKNOWLEDGMENT

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).

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] G. Ibanez et al. “ARP Path: ARP-based Shortest Path Bridges”. IEEE Communication Letters July 2011

[3] Ibanez G. et al. ARP-Path Ethernet Switching: On-demand Efficient Transparent Bridges for Data Center and Campus Networks. LANMAN, May 2010

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

[5] Mininet. McKeown Group Wiki http://yuba.stanford.edu/foswiki/bin/view /Main/ WebHome

[6] Soekris. http://www.soekris.com

[7] OpenFlow Project: http://www.openflowswitch.org

[8] Albert Greenberg et al. VL2: a scalable and flexible data center network. In Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09). ACM, New York, NY, USA, 51-62.