Evaluating Native Load Distribution of ARP-Path Bridging Protocol in Mesh and Data Center

Guillermo Ibáñez, Juan A. Carral, Elisa Rojas, Jose Manuel Giménez-Guzmán

Abstract **ARP-Path is a simple, low latency, shortest path bridging protocol for campus, enterprise and data center networks. We recently found that this protocol natively distributes the traffic load in networks having redundant paths of similar characteristics. The reason is that every new path between hosts is selected on-demand in a race among ARP Request packet replicas over all available paths: the first arriving replica gets its path selected on the fly. This means a continuous adaptation of new paths to variations on the load at links and bridges. To show this unique load distribution capability and path diversity property we use a number of simulations for complex scenarios, including two different simulators: one flow-based and one packet-based, and two basic topologies: data center and a regular mesh. We also verify this behavior on real hardware on a network of nine ARP-Path NetFPGA switches. The conclusion is that the ARP-Path protocol efficiently distributes traffic via alternative paths at all load levels, provided that multiple paths of similar propagation delays are available.**

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

I. Introduction

Ethernet switched networks are today the indisputable choice in terms of price/performance ratio, compatibility and zero configuration. However, the spanning tree protocol (STP) severely limits the performance and size of Ethernet networks. Standards like Shortest Path Bridges (SPB) [1] and Routing Bridges (TRILL) [2] respond to the need of adapting Ethernet switches to provide single IP subnets in campus networks while allowing utilization of all infrastructure links to obtain shortest paths. SPB and TRILL Rbridges use a link-state routing protocol (IS-IS), operating at layer two, to obtain shortest paths between bridges. These paths are used by all the flows that share the same source and destination bridges. To perform some load balancing, Equal Cost Multiple Path (ECMP) algorithms are implemented. This means additional complexity in terms of computation cost [3]. ECMP also may disturb traffic from other sources.

We proposed and implemented [4-5] ARP-Path Ethernet switching (also known as FastPath): a low latency and zero-configuration protocol for campus, enterprise and data center networks that enables the use of all available links without link state routing. ARP-Path operates as an *on-demand* bridging protocol that uses the standard, unmodified ARP frames to set up paths between hosts when the path is needed. ARP-Path protocol is an evolution of the transparent bridging paradigm that finds the shortest paths by flooding native ARP broadcast frames. It belongs to a new category of bridges, *All-Path*, because all possible network paths are simultaneously probed before its selection. Moreover, a new mechanism for locking the learning of source addresses provides native prevention of frame loops and allows utilization of all infrastructure links.

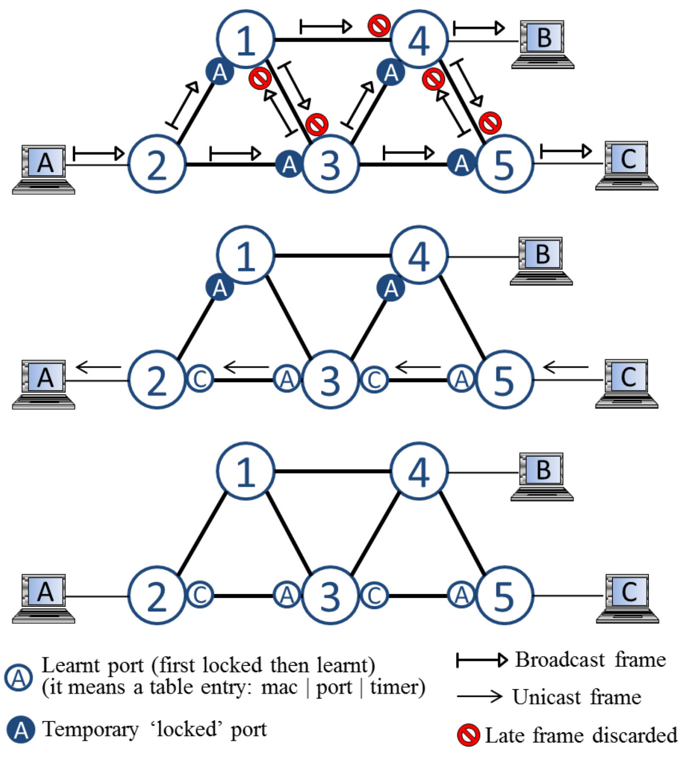
Unlike other proposals like SPB and TRILL, that compute the shortest paths between bridges and require complex algorithms to distribute the load, ARP-Path protocol automatically achieves an efficient load distribution across redundant links in a simple way regardless of the network load. This derives from the basic concept of the protocol: on-demand selection of the lowest latency path on a per host basis. In this paper we study this load distribution behavior.

The rest of the paper is organized as follows. In Section II we describe the ARP-Path protocol and a model of its behavior. In section III we tackle the problem with two simulators and a NetFPGA implementation. Section IV is devoted to an analytic solution of the load distribution of ARP-Path. Section V contains acknowledgments and Section VI contains the conclusions.

II. ARP-Path Protocol

The path set up in ARP-Path protocol [4] is performed by fully flooding the standard ARP Request frame sent by the source host, by *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. As shown in Fig. 1, host A sends an ARP Request packet encapsulated into a broadcast frame *b* to resolve the IP address of destination host C. Every bridge forwards *b* to all ports except the one through which it was first received and associates the global MAC address of A to this port, temporarily locking the learning of A's address to this port and blocking all other ports from learning and forwarding further received broadcast frames from source address A. When the ARP Request frame reaches host C, it responds with an unicast ARP Reply towards A that when snooped at every bridge (5,3,2) confirms the path.

The way that ARP-Path protocol is able to set up a path can be modeled as follows. New flows, that arrive to the system at mean rate λ and request a holding time with rate μ, are routed to any of the possible paths P_j, being N the number of possible paths ($P_1,..,P_N$) between the origin and destination. To model the scheduling policy of the ARP-Path bridging protocol we consider that an arrival flow selects the path with the lowest latency (ties are broken with equal probabilities). The latency



Learnt port (first locked then learnt) (it means a table entry: mac | port | timer)
Temporary 'locked' port
Broadcast frame
Unicast frame
Late frame discarded

Figure 1.Path set up for hosts A and C with ARP Request and Reply.

of a path can be computed as the sum of the latencies of all links of a path. Note that a link can belong to several paths simultaneously. We assume that there is a relationship between the latency of a link and a cost function, that defines the cost of link *i* at time *t*:

$$c_i(t) = \left(\frac{1}{1-u_i(t)}\right)\frac{\omega}{L_i}$$

being ω a constant that marks a reference cost (e.g. 10000), L_i the capacity of link *i* (Mbps) and $u_i(t)$ the link *i* utilization at time *t*, i.e. $u_i=l_i(t)/L_i$, being $l_i(t)$ the occupancy of link *i* at time *t*. Note that by using the above mentioned cost function we are modeling the delay in a link as an exponential function of the link load [6, Ch. 3]. Being the total cost of path *j* the sum of the costs of all the links that define the path

$$C_j = \sum_{\forall i \in P_j} c_i(t),$$

in our model, ARP-Path chooses the path *j* with minimum cost, i.e.

$$min_{j\in N} C_j = min_{j\in N} \sum_{\forall i \in P_j} c_i(t).$$

III. ARP-Path Flow and Packet Simulators

The model described in Section IV below becomes analytically intractable even for simple networks. Therefore we rely on network simulation to characterize the protocol in more complex networks. We follow two basic approaches: a packet-based simulator for maximum accuracy at the cost of long simulation times (implemented with OMNeT++) and a flow-based simulator with coarser granularity but capable of analyzing bigger scenarios with higher statistical accuracies (implemented with Simpy). As each simulator offers us some different benefits as explained before, we have used both to show the load balance capabilities of ARP-Path. Both simulators use the same traffic model, taken from [7]-[8]. A single flow generator installs new flows in the network according to a Poisson arrival distribution of rate λ. Every flow carries on average 34.8 MB of data (following a truncated Pareto distribution between 8MB and 8GB) at 0.5 Mbps (30% of flows), 1 Mbps (60%) or 10 Mbps (10%), and it is assigned to a given pair of network source and destination nodes according to a given traffic matrix derived from a gravity model. In order to assess the load adaptive capabilities of ARP-Path, we choose to simulate a simple and regular network, a 3x3 mesh, which offers many paths of similar costs between given pairs of nodes (see Figs. 2 to 7). For instance, there are six paths (of four hops each) spanning from node 0 to node 8 (0-1-2-5-8, 0-1-4-5-8, 0-1-4-7-8, 0-3-4-5-8, 0-3-4-7-8 and 0-3-6-7-8).

A. Flow-based simulator

We have implemented a simulator in Python (Simpy Lib) [9] to thoroughly evaluate ARP-Path. It uses a flow-based approach instead of a packet-based one according to the quasi-static assumption stated in [10, Sect. 5.4]. Every new flow is assigned a rate and volume according to the traffic model, a pair (source, destination) according to the traffic matrix and a route computed by applying the Dijkstra shortest path algorithm and the cost (latency) model shown in Section II.

We use two basic scenarios: one with uniform traffic matrix (with low, medium and high traffic load) and other with traffic matrix biased towards node 6 by a gravity factor of 4. This helps to visualize the deviation effect of more intense traffic at node 6.

To show the load balance capability of ARP-Path we have used one main metric, the link load utilization because it shows the balancing capability in a very simple and intuitive manner. However, we have also added some results that include the number of flows that traverse every link because it can give another insight to understand the load balance capability. The following Figs. 2 and 3 show the average results of 10 runs (128000 seconds each run) in a 3x3 grid network (1 Gbps link rate) and flow inter arrival times (1/λ) of 0.16, 0.04 and 0.024 seconds. At Fig. 2 the link load utilization, i.e. the average link load in Mbps, is shown under every link (in both directions). The overall network load is well balanced across all links. The bottom right diagram in the figure shows the scenario with higher traffic from/to node 6.

In order to get a better insight on the distribution of load between two edge nodes, we selected, among all the traffic, the flows between node 0 and 8 and registered their selected paths at bridges. Figure 3 shows the percentage of flows between nodes 0 and 8 that traverse each link. It can be seen how the flows from node 0 to 8 are balanced between the available routes. At bottom right, with traffic matrix biased toward node 6, it can be seen how flows from 0 to 8 are pushed away from node 6 (lower right) when links around node 6 carry more traffic due to the higher traffic at node 6.

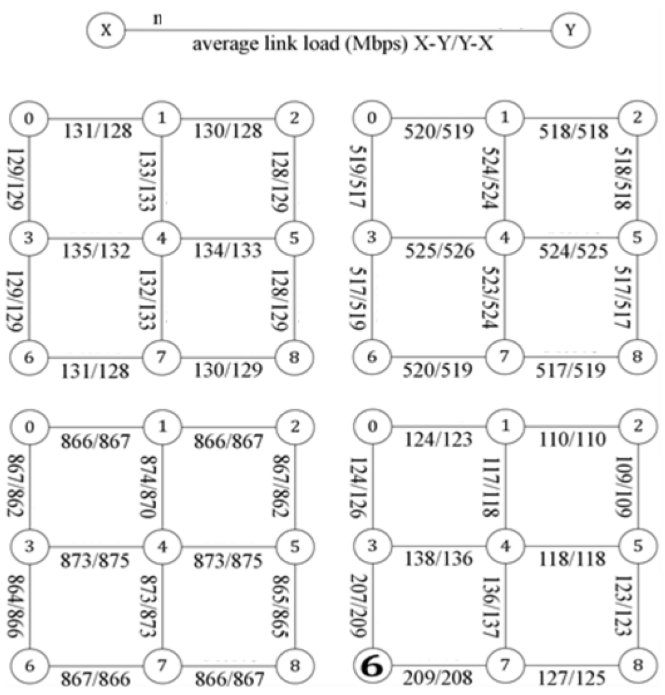


Figure 2. Load distribution with the flow-based simulator under uniform traffic matrix in scenarios with different load: low (upper left), medium (upper right) and high (lower left). On the lower right a traffic matrix biased towards node 6 by a gravity factor of 4 is used.

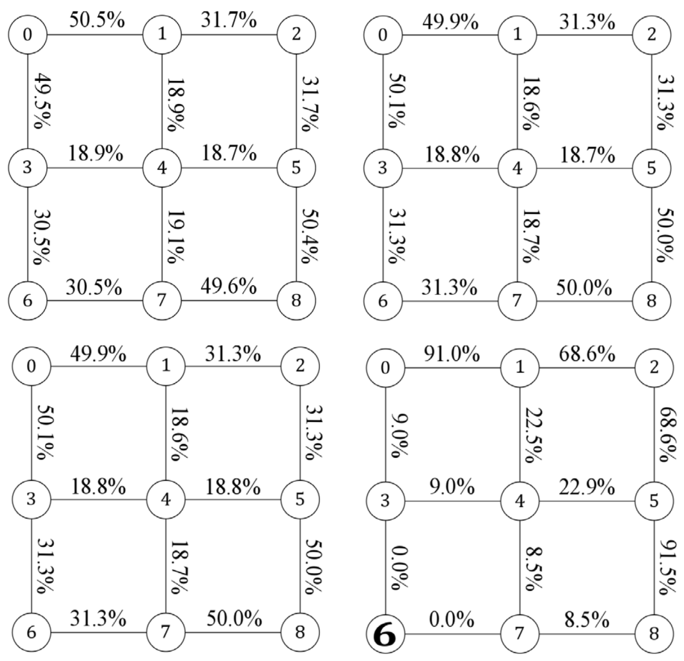


Figure 3. Distribution of the flows between node 0 and 8 with uniform traffic matrix and under a traffic matrix biased towards node 6 by a gravity factor of 4 (right) in the flow-based simulator.

The results of the flow-based simulator show a very good load distribution, but the flow model does not deal with packets. In order to get a more realistic performance evaluation, we implemented the same traffic model on a packet-based simulator: OMNeT++, but simulation times are several orders of magnitude bigger than with Simpy.

B. Packet-based simulator

The packet simulator has been implemented in OMNeT++ [11] and using the INET library [12]. Figures 4 to 6 show the average results of two runs (10000 seconds each run) in a 3x3 grid network (100 Mbps link rate) and flow inter-arrival time (1/λ) of 1.6 seconds. To show the protocol load distribution capabilities, the percentage of flows originated at node 0 and destined to node 8 (edge to edge) and the total link load utilization (in each direction) is provided for each link. Figure 4 shows how the flows from node 0 to node 8 are evenly distributed (by halves at every node) among the four main paths spanning from bridge 0 to bridge 8 when no other traffic is exchanged in the network.

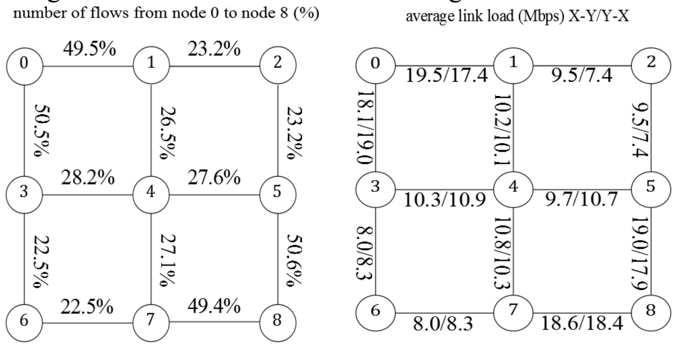


Figure 4. Load and flows distribution (nodes 0 to 8) without background traffic in the packet-based simulator.

Figure 5 shows the load at links with background traffic in two different scenarios: when a uniform distribution of traffic between all source and destinations is used (left) and with traffic biased towards node 6 by a factor of 4 (node 6 has four times more probability to be chosen as source or destination than any other node).

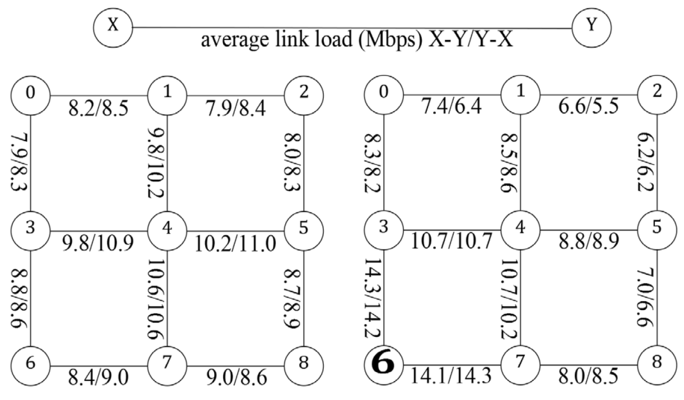


Figure 5. Load distribution with the packet-based simulator under uniform traffic matrix (left) and under a traffic matrix biased towards node 6 by a gravity factor of 4 (right).

Figure 6 shows the percentage of flows between nodes 0 and 8 at every link with good load balancing, although not as precise as with the flow model, due to the fact that the simulation times are more than 10 times smaller. The mesh on the right shows how the protocol adapts to the traffic conditions diverting some flows from 0 to 8 to paths away from node 6.

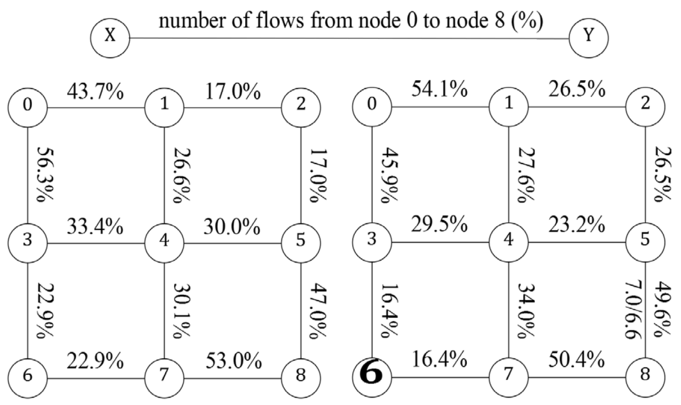


Figure 6. Distribution of the flows between node 0 and 8 with uniform traffic matrix and under a traffic matrix biased towards node 6 by a gravity factor of 4 (right) in the packet-based simulator.

Notice that Figs. 5 and 6 are analogous to Figs. 2 and 3, the difference is the simulator used and the parameters and times of the simulation, smaller for the packet-based one, and the flow-based shows four different scenarios while the packet-based only shows two of those four. Figure 4 adds a different scenario for the packet-based simulator in which there is no background traffic exchanged but between nodes 0 and 8.

C. *NetFPGA ARP-Path bridges load distribution tests*

A 3x3 mesh of NetFPGAs available at Cambridge Computer Lab has recently been used to test path diversity with real ARP-Path bridges implemented on NetFPGA [13]. In this test, eight virtual machines on a PC connected to bridge 0 set sequentially a UDP video streaming traffic with the VLC media player [14] to other eight virtual machines connected to bridge 8. The selected paths are registered: path diversity is the norm. We reproduce at Fig. 7 a typical sequence of the eight paths selected for the eight video streams set up successively:

1st	Nodes 0-1-4-5-8
2nd	Nodes 0-3-4-5-8
3rd	Nodes 0-3-6-7-8
4th	Nodes 0-3-4-7-8
5th	Nodes 0-1-4-7-8
6th	Nodes 0-3-4-5-8
7th	Nodes 0-1-2-5-8
8th	Nodes 0-1-4-7-8

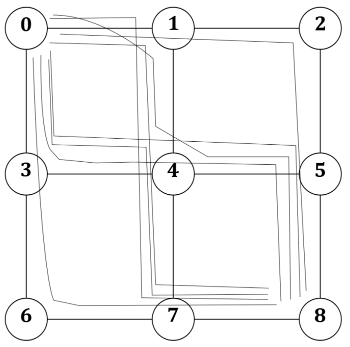


Figure 7. Example sequence of paths selected by eight independent video stream flows from virtual hosts connected at bridge 0 to hosts at bridge 8.

It is worth noting that NetFPGA links have a capacity of 1 Gbps, video stream represents a very small fraction of link capacity and ARP-Path still balances the load. Another fact observed on this network with NetFPGA cards (real hardware instead of simulations) is that, when there is no traffic at all, there is only one fastest path between two bridges that is always selected. While simulations choose randomly one path among the four possible in the previous topology (because they all have exactly the same latency when there are no packets in the network), real hardware cards behave with small but consistent differences, so even the smallest difference in latencies will cause the topology to have only one fastest path. In order to prove that load balancing was not random, but based on the path latency, a single flow from a PC at 0 to a PC at 8 was repeatedly selected without any other traffic at the network. After 16 repetitions (one after the other, always with empty network), the path chosen was always the same: 0-1-4-7-8, the fastest real hardware path. However, when multiple paths are set up together and they share links, the paths diversify and load is balanced, as shown in Fig. 7.

D. *Data center topology with packet-based simulator*

We also simulated the network shown in Fig. 8, frequently used in data centers. It consists of groups of 25 hosts connected to every ARP-Path top of rack switch (HS) and a core of four ARP-Path switches with redundant connections to them. All links have 100 Mbps capacity and 1 μs delay.

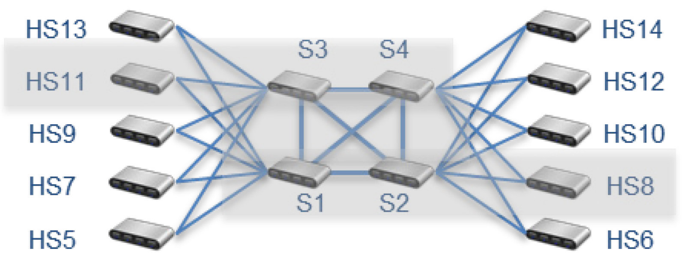


Figure 8. Topology used for alternative paths evaluation.

We focus on the shaded section HS11-HS8, a topology that is typical of fat tree networks if we eliminate the S3-S1 and S2-S4 links, which in practice carry no traffic between HS11 and HS8. Hosts on the left connected to bridge HS11 send UDP traffic to the hosts on the right connected to HS8. There is one traffic flow for each pair of hosts. UDP packets are of variable length and exponential inter arrival time of 1ms. Different runs with increasing message size, ranging from 50 bytes to 900 bytes, have been simulated so that the aggregated traffic in the core network section (going from the access to the core. The aggregated traffic injected to the core at HS11 is 20, 70, 130 and 175 Mbps. The numbers at links of Fig. 9 show traffic in Mbps (also equivalent to % of link capacity). Balance of traffic, among the two links from HS11 and also between the other two links to HS8, is coarse at low loads, but tends to level off with increasing traffic. The ratio of loads at the two links of HS11 (and HS8) varies from 3.46 at low load up to 1.35 at high load.

To evaluate the traffic distribution with more distributed traffics, we also simulated this datacenter topology with the same traffic model used in the 3 x 3 mesh above and with simulation times of 5000 seconds. All hosts have equal traffic weight. Figure 10 shows the percentage of routes from HS11 to HS8 used by the link. Figure 11 shows the average percentage of link utilization in both directions for four different load levels corresponding to inter arrival times of 0.4, 0.2, 0.1 and 0.05 seconds. The vertical links S3-S1 and S2-S4 are only used with higher loads for a small fraction of the routes when more direct routes become significantly loaded.

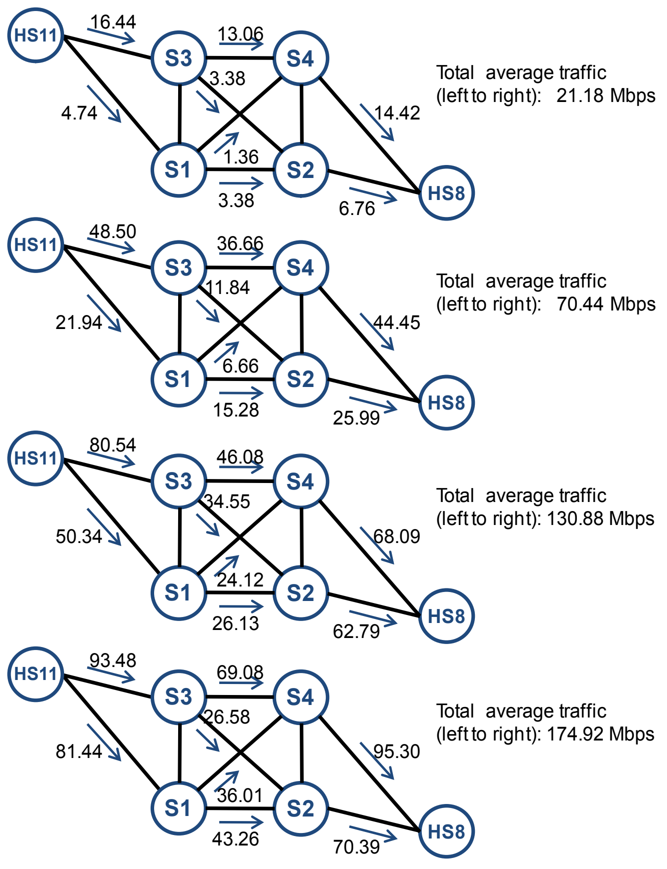


Figure 9. Traffic load links in % (also Mbps) for increasing total traffic injected from hosts at HS11 to hosts at HS8.

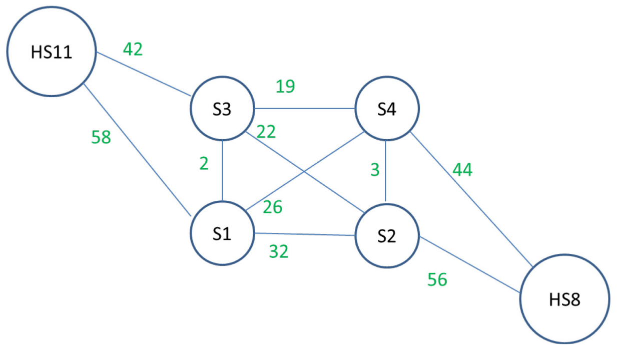


Figure 10. Percentage of routes from HS11 to HS8 traversing each link

IV. Basic Analytic solution

In this section we describe an analytic solution of the described system. The scope of this solution is rather limited, because, for the sake of mathematical tractability we consider the number of paths *N=2* and that each path is formed by a single link. Routing is performed by choosing the path with maximum available capacity, which is expected to have the minimum cost and delay, so the behavior of the system can be *described by a discrete-state continuous-time process*. We also make the common assumptions of exponentially distributed random variables for the inter-arrival and holding times of the flows with parameters λ and μ, respectively. Note that in the section devoted to simulations we have considered

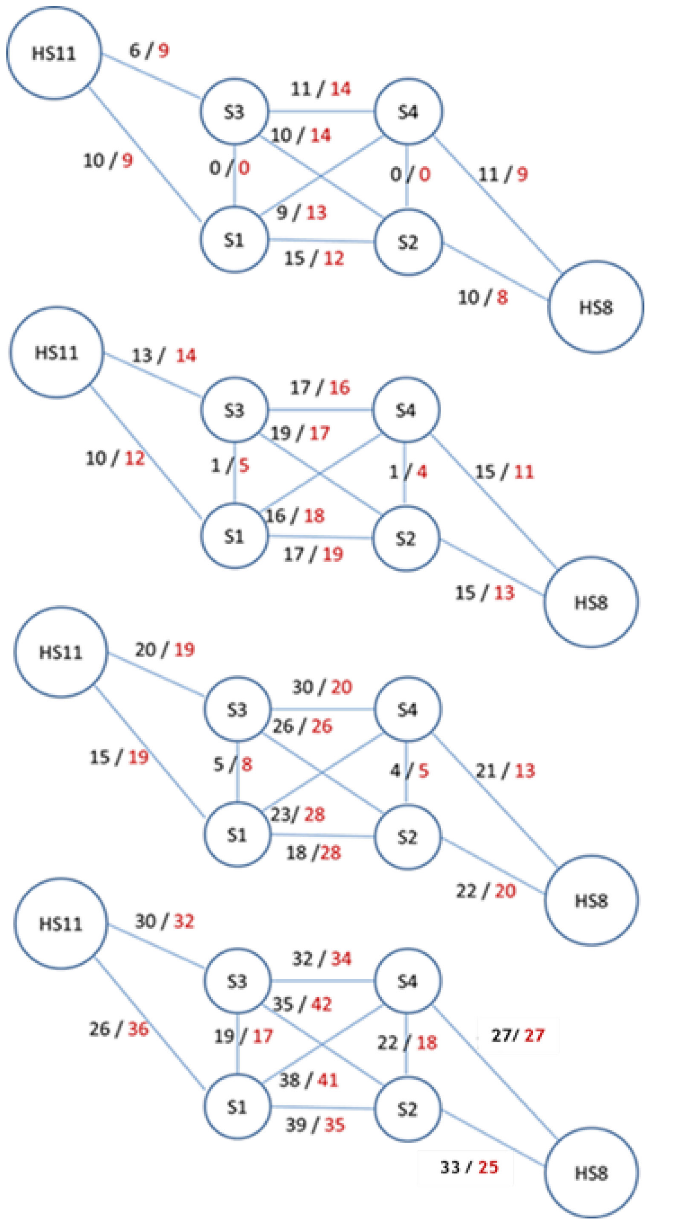


Figure 11. Link utilization in both direction in Mbps in section between nodes 11 and 8 with uniform traffic matrix at four different loads.

more realistic distributions. We can represent the state of the

$$\begin{tikzcd}[column sep=large, row sep=large]
 & (i\text{-}1,j) & \\
(i,j\text{-}1) & (i,j) \arrow[l, "\begin{cases}\lambda, & j>i \\ \lambda/2, & i=j \\ 0, & j<i\end{cases}"'] \arrow[u, "\begin{cases}\lambda, & i>j \\ \lambda/2, & i=j \\ 0, & i<j\end{cases}"'] \arrow[r, "(L_2\text{-}j)\mu"'] \arrow[d, "(L_1\text{-}i)\mu"'] & (i,j+1) \\
 & (i+1,j) &
\end{tikzcd}$$

Figure 12. Transition diagram.

system at any given time by a vector $S:=\{s_1,s_2\}:0\leq s_1\leq L_1;\ 0\leq s_2\leq L_2$, where s_i is the available capacity of path i ($0<s_i<L_i$), that is determined by $s_i(t)=L_i\text{-}l_i(t)$. Without loss of generality, we consider that each flow occupies one resource unit, so L_i in this section is measured in resource units. This system is therefore a CTMC whose transitions rates are depicted in Fig. 12 and constitutes a level dependent Quasi Birth and Death Process (QBD) [10] whose infinitesimal generator (***Q***) has the following block tridiagonal structure, being the block size $(L_2+1)\times(L_2+1)$:

$$Q=\begin{bmatrix} D_0 & M_0 & 0 & 0 & \dots & 0 & 0 & 0 \\ L_1 & D_1 & M_1 & 0 & \dots & 0 & 0 & 0 \\ 0 & L_2 & D_2 & M_2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & L_{L_1} & D_{L_1} & M_{L_1} \end{bmatrix}$$

The stationary probability distribution can be obtained by solving $\boldsymbol{\pi Q=0}$ along with the normalization condition. As ***Q*** is a finite matrix this system can be solved by any of the standard methods defined in classical linear algebra.

In Fig. 13 we show the main results obtained solving this model for μ=1 and for different values of the offered load to the system $\rho=\lambda/\mu$, so that the system operates in very different working points. On the left-hand side we consider a scenario with the same capacity for both links L_1=L_2=20, whereas on the right we show the results for a scenario with $L_1=30$ and L_2=20. These figures show the probability of having a different available capacity of Ψ resource units between the different paths, i.e. for Ψ=0 we show the probability of both paths having the same available capacity, and in $\Psi=i$ ($-i$) we represent the probability that path 1 (2) has i more available resource units than path 2 (1).

As it can be concluded from the figure, the probability of being in a state where there is a path with much more resources than the other (high values of $|\Psi|$) is negligible, so the load is properly distributed in order to get the highest available bandwidth (minimum delay).

V. Acknowledgment

This work was supported in part by grants from Comunidad de Madrid through Project MEDIANET-CM (S-2009/TIC¬1468) and CMACs (UAH2011/EXP-016). Thanks Andrew Moore and Jon Crowcroft for letting us using the Cambridge Computer Lab facilities for the NetFPGA tests.

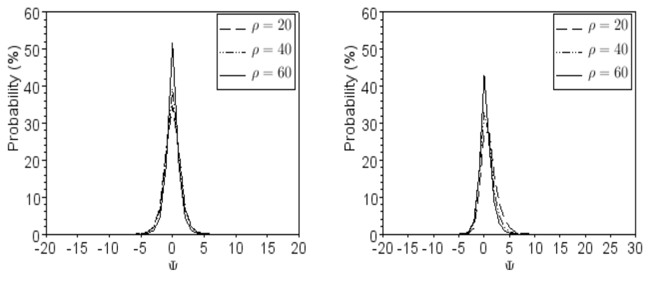


Figure 13.Probability of having a different available capacity (of Ψ resource units) between paths of equal capacity (left) and different capacity (right).

VI. Conclusion

ARP-Path protocol provides native load distribution in network topologies having redundant paths of similar characteristics, like fat trees and meshes. Experimental results from simulations and implementations show consistent load distribution across different scenarios and traffic loads in the topologies studied, where multiple similar paths are available like data center topologies with fat tree structure and regular topologies. Further study is needed to fully characterize the boundaries and limitations of load adaptive routing in networks with redundant paths of dissimilar characteristics.

VII References

[1] Shortest Path Bridging. http://www.ieee802.org/1/pages/802.1aq.html
[2] Transparent interconnection of lots of links (TRILL) WG. http://www.ietf.org/html.charters/trill-charter.html
[3] J. Farkas, Z. Arató. "Performance analysis of shortest path bridging control protocols". GLOBECOM 2009 Honolulu pp. 4191-4196.
[4] G. Ibáñez, J.A. Carral, J.M. Arco, D. Rivera, and A. Montalvo. "ARP-Path: ARP-Based, Shortest Path Bridges". IEEE Communications Letters, 2011, pp.770-772.
[5] E. Rojas. J. Naous, G. Ibáñez, D. Rivera, J.A. Carral, J.M. Arco. "Implementing ARP-Path Low Latency Bridges in NetFPGA". SIGCOMM 2011 demos. August 2011.
[6] D. Bertsekas and R. Gallager. "Data Networks". Prentice-Hall, New Jersey, 1987.
[7] R.S. Prasad, C. Dovrolis. "Beyond the Model of Persistent TCP Flows: Open-Loop vs Closed-Loop Arrivals of Non-persistent Flows". anss-41, pp.121-130, 41st Annual Simulation Symposium (anss-41 2008), 2008.
[8] A. Kvalbein, C. Dovrolis, and C. Muthu. "Multipath load-adaptive routing: Putting the emphasis on robustness and simplicity". In Proceedings of ICNP, Princeton, New Jersey, USA, October 2009, pp. 203-212.
[9] Python Simpy: http://simpy.sourceforge.net
[10] M. F. Neuts. "Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach" vol. 2 of Johns Hopkins Series in the Mathematical Sciences, Johns Hopkins University, Baltimore, Md, USA, 1981.
[11] OMNeT++ Network Simulator Framework: http://www.omnetpp.org
[12] INET Framework: http://inet.omnetpp.org
[13] NetFPGA: http://netfpga.org
[14] VideoLAN VLC media player: http://www.videolan.org/vlc