2012 IEEE Global Communications Conference (GLOBECOM) (2012). (pp. 2523 - 2528). IEEE
DOI: http://dx.doi.org/10.1109/GLOCOM.2012.6503496

Torii-HLMAC: A Distributed, Fault-tolerant, Zero Configuration Fat Tree Data Center Architecture with Multiple Tree-based Addressing and Forwarding

Elisa Rojas
Departamento de Automatica, UAH
Alcala de Henares (Madrid), Spain
elisa.rojas@uah.es

Guillermo Ibáñez
Departamento de Automatica, UAH
Alcala de Henares (Madrid), Spain
guillermo.ibanez@uah.es

Abstract— **This paper describes Torii-HLMAC, a scalable, fault-tolerant, zero-configuration data center network fabric architecture protocol as a fully distributed alternative to PortLand for similar multiple tree network topologies. It uses multiple, fixed, tree-based positional MAC addresses for multiple path and table-free forwarding. Addresses are assigned by a simple extension of the Rapid Spanning Tree Protocol. Torii-HLMAC enhances the PortLand protocol advantages of scalability, zero configuration and high performance and adds instant path recovery, fully distributed routing and address assignment. ARP Proxy may be used to avoid ARP broadcast messages.**

Keywords- Ethernet; tree-based routing; routing bridges; data centers; fat trees; Shortest Path Bridges; Spanning Tree.

I. Introduction

Data center networks are increasingly relying on Ethernet and flat layer two networks due to its excellent price and performance ratio and configuration convenience. The growing importance, by economic reasons, of the scale up model (increasing switches and links speeds) by the scale out model [1], using a high number of commodity servers and switches, is driving data center networks to high scale dimensions. Different approaches to implement a data center fabric have been recently proposed to overcome the limitations of Spanning Tree protocol (ST) and the configuration complexity of Multiple Spanning Tree Protocol.

PortLand [2] is a recent protocol proposal for data centers that uses centralized control, location based pseudo MAC addresses and Up/Down turn prohibition to prevent loops. Positional addresses are assigned to hosts and switches by a discovery protocol and replace universal MAC addresses at edge switches. ARP proxy and path repair functions are implemented as centralized functions. Previous to PortLand were [3] and [4] (SEATTLE).

In this paper we explore a combination of distributed functions to make forwarding in fat trees simple and more scalable. The Torii-HLMAC protocol aims to improve PortLand (and routing in fat trees in general) with alternative, simpler and distributed mechanisms. We use topological pseudo MAC addresses, but multiple simple addresses (inspired in TRE [5]), in order to facilitate multipath forwarding, direct frame routing without tables and on the fly alternate path selection after link failure.

In Section II, the basics of Torii-HLMAC will be explained: address assignment, broadcast/multicast/unicast forwarding and path repair. The topology used is the same shown in PortLand [2] for its simplicity. Nevertheless, in Section III more specific features of Torii will be detailed, as well as the generalization to any other topology.

II. Protocol Description

A. *Tree-based Multiple Addresses structure and automatic assignment with Extended RSTP*

Torii-HLMAC assigns to each bridge a Hierarchical Local MAC (HLMAC) address at every port connected upstream as shown in Fig. 1. Upper layer bridges are assigned one HLMAC, next layer bridges get two HLMAC addresses (one per link to upper layer bridges) and so on in powers of two.

HLMAC addresses are local (private) MAC addresses, i.e., addresses whose U/L bit is set to 1. The 46 bits available for addressing purposes (after removing the local or global bit and the multicast bit), encode by default up to 6 different hierarchical levels, with 6 bits for the first and 8 bits for each other level. The HLMAC of a bridge is expressed in the dotted form **a.b.c...** as the chain of designated port IDs a, b, c, ... traversed in the descending path from the Root Bridge to the bridge to which the address is assigned.

To build the spanning tree and assign hierarchical addresses to the bridges, a modified version of the Rapid Spanning Tree Protocol (RSTP) is used, which is defined in HURP [6]. Once the root bridge is set, which gets 0.0.0.0.0.0 as HLMAC, the process of building the spanning tree from the root to the leaves starts. This iterative process consists of BPDUs being sent by the parent bridge including the number of the Designated Port. These numbers are 1,2,3,4, which correspond to the pod number that the port is connected to, as shown in Fig 1.

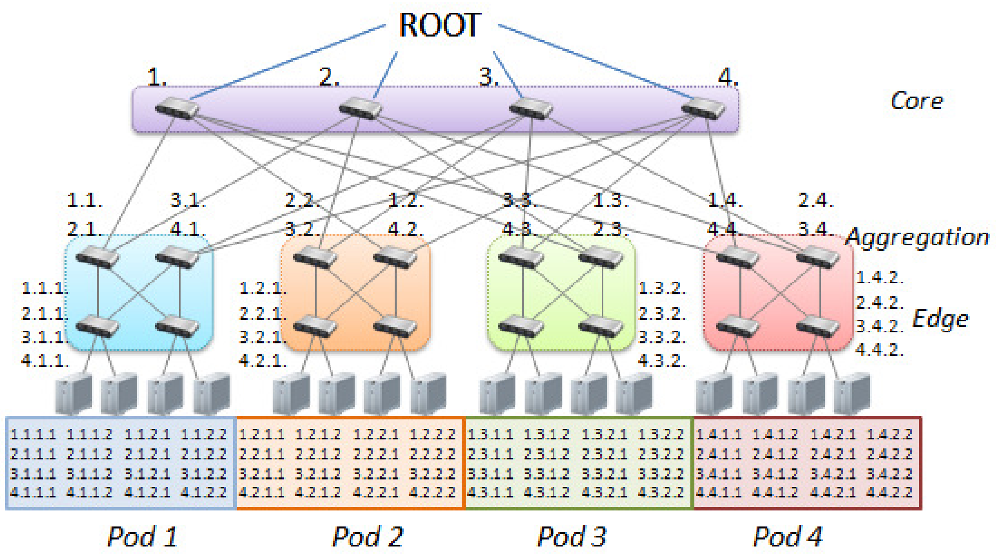

Figure 1: Multiple hierarchical addresses (HLMAC) assignment for Torii with extended Rapid Spanning Tree Protocol from virtual Root node. From '.' it means the rest of the HLMAC is made of zeroes.

For instance, the first switch at pod 1 has 1.1. (core switch 1 and designated port 1) and 2.1. (core switch 2 and designated port 1) as its HLMAC addresses.

The role of root bridge may be implemented as a couple of root bridges operating in active and fully sharing its addressing system towards downstream ports with redundant links, so that bridges 1,2,3,4 always get the same address via both links, even if a root bridge links fails. Root bridges do not carry user traffic because the reflection plane is located at core bridges, then their design is only focused to dependability.

As a result, each node gets one or more (up to four in this scenario) topological tree addresses, HLMAC. There will be so many alternative HLMAC addresses at the edge switches as the number of core switches, and the HLMAC prefix will be used to distribute traffic on a hash base.

B. *Tree-based Forwarding*

Routing of every frame is directly performed via address decoding. Once the HLMACs are set, Torii switches need to distinguish among broadcast/multicast and unicast frames, and identify the direction of the frame: "going up" or "going down", which is done thanks to the frame input port. Once those two parameters are known, the logic applied at each switch of the topology is the following:

```
If frame is BROADCAST or MULTICAST:
  If frame goes UP:
    If switch is edge → host MAC to HLMAC (prefix chosen by a hash)
    Forward frame through the HLMAC port *
    Down broadcast frame ***
  Else if frame goes DOWN:
    If switch is edge → HLMAC to host MAC
    Down broadcast frame ***
Else if frame is UNICAST:
  If frame goes UP:
    If switch is edge → host MAC to HLMAC (prefix chosen by a hash)
    Forward frame through the HLMAC port **
  Else if frame goes DOWN:
    If switch is edge → HLMAC to host MAC
    Forward frame through the HLMAC port *
```

**: Forwards through the next port according to the HLMAC address (up port if the frame comes from a down one or viceversa)*
***: Same that *, but in unicast, sometimes frames do not need to reach the core switch if there is a shorter path, in this case the frame is not forwarded to the core (indicated by the HLMAC prefix).*
****: Broadcast only through the ports located down in the hierarchy except through the input port.*

1) *Broadcast and Multicast Forwarding*

To put it into words, when a host A sends a broadcast frame, first of all, the switch serving that host chooses a prefix based on a hash function. The prefix determines the core switch that will be used to carry out the broadcast. For instance, if prefix 1 is chosen, while the broadcast destination address (FF:FF:FF:FF:FF:FF) remains the same, the frame source address A is translated into the corresponding Torii-HLMAC address (see Fig. 2, in which A source address is translated into 1:1:1:1:0:0 by the edge bridge). Broadcast frames from a specific host may use different prefixes to obtain load distribution and path diversity, since the hash function can be based on any flow-related parameter.

Once the prefix is elected and the address translation from global MAC to local HLMACs is done, the frame is directed up to the matching core switch and also replicated down to every link except the one associated to the input port as shown in Fig. 2. Since only one core switch is used, paths are unique and the communication remains loop-free.

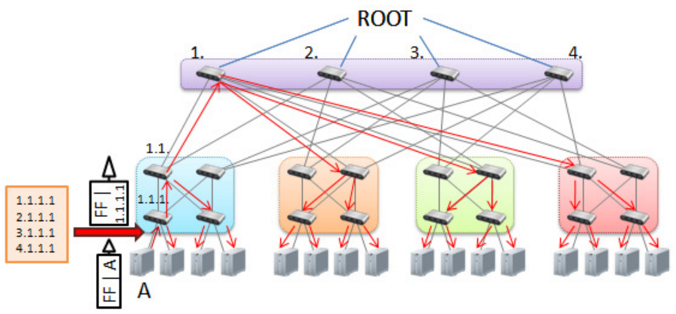

Figure 2: Broadcast frame from host A. The broadcast address remains the same while the A source address is translated into 1.1.1.1 at edge bridge in the frame when prefix 1 has been chosen at the edge by hash.

Finally, if the received frame at the destination edge switches is an ARP Request, the chosen prefix HLMAC is reversed into A (the information is known thanks to the ARP message) and both addresses (the HLMAC and the original MAC address A) are saved in a table for future unicast frames.

The same forwarding mechanism is applied for multicast frames. As it can be seen, multicast and broadcast forwarding are performed across the spanning tree as it occurs in classical Ethernet, the differences are the multiple (four in the figure) trees available where the frame is forwarded only to one of them by selecting the first level bridge with the hash function. ARP Proxy [7] function may be implemented at edge bridges, learning from all ARP Request and ARP Reply frames traversing them, or centralized, as in PortLand.

2) *Unicast Forwarding*

In the case of unicast frames, the hash to select the prefix is based on both the origin and the destination MAC addresses, so that the prefix is unique and the flow is guaranteed to be bidirectional. Additionally, some other parameters (for instance, transport layer parameters such as the protocol or the transport ports) might be added to the hash function in order to distribute the traffic more efficiently, since multiple paths between hosts can be generated.

Once the unicast frame arrives at the source edge switch, it translates both addresses. The origin address is translated into the corresponding Torii-HLMAC address (which is known by the edge switch, since it is the responsible of assigning it to its hosts) and the same happens with the destination address (its HLMAC is always known by a previous ARP Request, which will be always sent before any unicast frame).

For instance, in Fig. 3, prefix 1 was chosen and, because of this, the origin B is translated into 1:3:1:2:0:0, while the destination A is translated into 1:1:1:1:0:0, which is known by the previous ARP Request (see Fig. 2).

If a different prefix was chosen with the ARP Request, the according Torii-HLMAC would be easily deduced. If prefix 2 had been chosen instead, B would be 2:3:1:2:0:0 and A, 2:1:1:1:0:0. Therefore only the prefix part of the HLMAC changes, an advantage of the fixed topology, which makes HLMAC addresses completely deducible once one single HLMAC is already known.

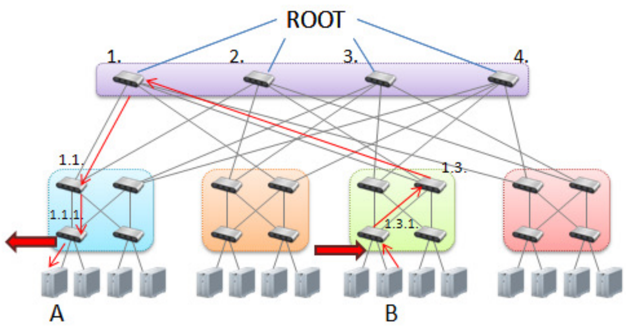

Figure 3: Unicast frame from B to A. Both addresses (A and B) are translated at the edge switches, which already know them from the previous ARP messages. In this case A goes 1.1.1.1 and B goes 1.3.1.2.

After the prefix is elected and the translation done, the frame is forwarded up or down according to the destination HLMAC. The main difference with the broadcast forwarding is that the frame does not always need to travel to the core switch to finally reach the destination, because, sometimes, there will be shorter paths, for example if the hosts share the edge switch or the pod.

Finally, at the destination edge switch, if the frame is an ARP Reply, both addresses (the destination MAC and translated HLMAC) are saved in a table. In this way, thanks to the ARP Request and Reply messages, origin and destination edge switches will be capable of translating future unicast frames.

Notice that unicast frames will always have the same prefix in their Torii-HLMAC destination and source addresses, this is because they will always travel through the same core switch so that the communication is bidirectional. Therefore, unicast frames with different prefixes at their Torii-HLMACs can be used for special events such as notifying an action (for example, a failed link when found) to any switch in the topology.

C. *Tree-based Path Repair*

In the previous section, standard forwarding has been detailed. When a link goes down in the topology, no messages are exchanged and the Torii-HLMACs assigned remain exactly the same, only the switches connected to that link know that the link is down and not valid for a path anymore.

Therefore, when a frame arrives at a switch and its Torii-HLMAC indicates it should be sent through the failed port, the path repair procedure starts. The method is as follows, an alternative HLMAC can be directly deduced on the fly from its HLMAC. For this deduction is necessary to consider the following:

- *Broadcast and Multicast frames:*

  As there are no needs to follow a specific path, and to prevent multiple broadcasts, the first and closest alternative path available is selected. By closest, we mean the alternative that requires fewer steps back to continue the broadcasting.

- *Unicast frames:*

  Since the communication must be bidirectional, both sides of the communication should take the same alternative path, i.e. the same alternative core switch as the prefix. The criterion in this case is to choose the next core switch in order, even if it is not the closest alternative.

  A notification should also be made to the edge switches to avoid selecting the non-valid prefix path again.

Once the alternative HLMAC is selected, the frame is forwarded back (if necessary) until it reaches the new path and then it is transmitted through it as shown in Fig. 4.

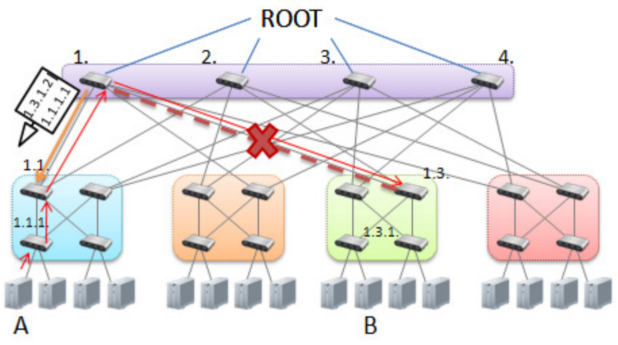

Figure 4: Unicast frame from A to B. Link from switch 1. to 1.3. is down, when the frame arrives at 1. it is forwarded back to 1.1. to be sent to its new alternative path, which is the one with the next prefix, 2.

In the case of unicast frames, since the edge switches need to be notified, the source HLMAC will be translated into the new HLMAC while the destination will remain the same. In this way, the same redirected frame serves as a notification as well, because their HLMAC prefixes will not be the same and it will be considered a special frame.

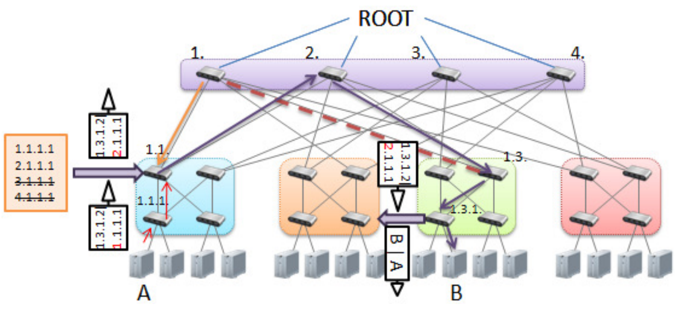

Figure 5: Unicast frame from A to B. Once the frame is back to 1.1., the origin HLMAC is translated into the new path HLMAC (from 1.1.1.1 to 2.1.1.1) and sent to core switch 2. The destination remains the same, 1.3.1.2, but because of its nature (different HLMAC prefixes), it is known the frame is a notification frame and it is sent to the new HLMAC: 2.3.1.2.

The notification indicates the switch serving the destination host to avoid using the old path (shown in the destination prefix) and start using the new one (shown in the source prefix). For instance, in Fig. 5, the source is translated into 2.1.1.1, while the destination remains as 1.3.1.2. However, since it is a failure notification frame, it is deduced to be sent to 2.3.1.2 through core switch 2, i.e., the frame does not follow the path of destination for routing, but the destination with the prefix of the source. As an example, in Fig. 5, the frame destination is 1.3.1.2 but it will be forwarded through 2.3.1.2 (core switch 2) since the source prefix is 2 and not 1, and at the same time, the frame indicates the failed link for core switch 1 to the destination edge switch.

Finally, once the frame arrives at the destination edge switch, this last sends the notification back so that the origin edge switch is also informed. This second notification has now the old origin as its destination and the new destination as its origin. For instance, in Fig. 6, the frame HLMAC origin is now 2.3.1.2 (from the previous destination 1.3.1.2) and the destination is 1.1.1.1 (from the previous source 2.1.1.1) but, since it is a notification, the frame is sent through the path to 2.1.1.1.

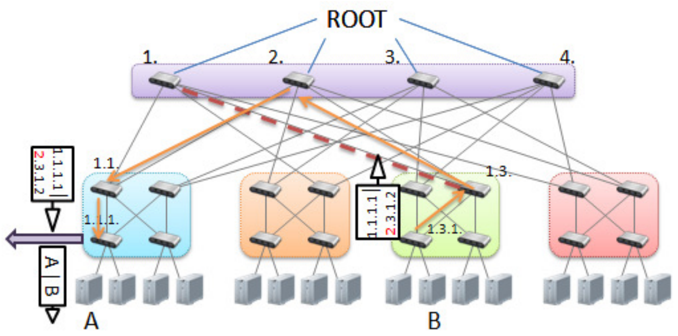

Figure 6: Unicast frame from A to B. Once the frame has notified the destination edge switch (1.3.1), it is sent back to 1.1.1.1 (through the new path 2.1.1.1) indicating the new destination HLMAC 2.3.1.2

After both notifications are exchanged, both edge switches know about the failed link and will not assign again that path after hashing, but the alternative one.

It is important to notice that these special frames are not extra messages in the network, they are just normal communication frames that serve, as the same time, as notifications. Therefore, when path repair is needed, not only alternative paths are decided on the fly, but there is no overload in the network.

III. Evaluation

A. *Simulation of Torii HLMAC*

Torii-HLMAC has been simulated in Omnet++ (v4.1). The implementation, coded in C++, relies on the MACRelayUnit module (from inet/linklayer/etherswitch). The base has been modified so that it acts as a Torii switch. Though STP is not implemented, the STP BPDUs are given as parameters in the simulation instead.

The simulations consisted of UDP traffic exchange between different hosts in the PortLand topology. Operation of path repair was also proven with broadcast and unicast traffic by changing the status to down of links at different hierarchy levels. As expected, frames were directly forwarded through the alternative path with no need of extra messages or any other calculations.

The aim of these simulations was to prove Torii-HLMAC works as expected: Once the HLMAC are assigned, no overload was added and decisions were made on the fly (for normal forwarding and repair as well), no extra messages were needed for any change of configuration, load balancing was assured with the multiple addressing and, for all of this features, we only implemented the logic of Torii-HLMAC at every switch. Regarding table sizes, switches only needed to save their addresses and edge switches require an extra table for saving the original host' MACs for translation of maximum size equal to the number of hosts in the network.

B. *Use of Virtual Machines at hosts*

In data center topologies, physical hosts are usually composed by a number of virtual machines (VM) installed on them. In the most generic case, Torii would assign one HLMAC address per final host, but in practice, Torii only uses the first four bytes of HLMAC addresses, so the last two bytes could be used to distinguish among those VM by being assigned in the reception order of their ARP messages. Therefore, every host could have up to 2^{16} (65536) – 1 (since number 0 is not used for HLMAC addresses) = 65535 VM working at the same time.

C. *HLMAC Address Assignment Alternatives*

In this proposal, Torii-HLMAC takes 1 byte of the 6 of the HLMAC per hierarchical level, so that is 4 bytes and the free 2 bytes could be used per VM addressing assignment. Nevertheless, if more levels are needed, fewer bits could be assigned per level and many alternatives could be used depending on the topology requirements, without changing the basics of the Torii-HLMAC protocol.

D. *Inter-L2 Mobility*

Regarding inter-L2 mobility, when a host (or virtual machine in a host) A communicating with another B, moves from one edge switch to a different one, the frames will follow the next procedure:

- If the frame goes from A to B (Fig. 7):

  If the new edge switch has B's HLMAC, the frame is forwarded towards it, with the new A's HLMAC, since we consider A should had emitted a gratuitous ARP before.

  If not, the edge switch should emit a special frame (ARP Request) to obtain B's HLMAC and discard any frame meanwhile. A second option would be broadcast any frame following the ARP-alike frame, so that they are not lost.

- If the frame goes from B to A:

  In this case, the frame will reach the old edge switch and this last should broadcast the frame towards the other edge switches (all frames so that they are not lost or just a single one to make the notification). The new edge switch would then note down B's HLMAC and send a special message (ARP Reply) towards B with the new A's HLMAC.

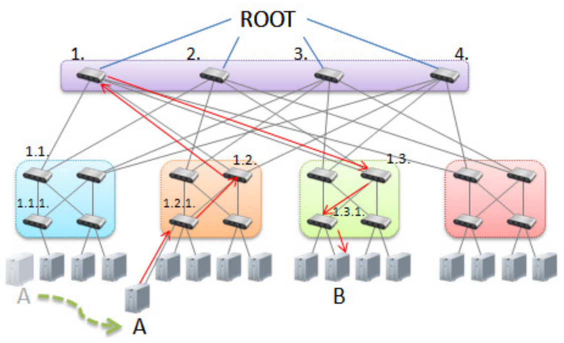

Figure 7: Unicast frame from A to B, after A change. The frame reaches its new edge switch and this already knows B (from a previous ARP or after an extra ARP by the edge). A is translated into 1.2.1.3 and reaches B at 1.3.1.2. The switch serving B knows A's address and can translate back thanks to it gratuitous ARP (or the ARP Request from A's edge switch if needed)

Both cases require broadcasting of frames in order to be lossless. However, alternatively, less broadcasting could be applied by discarding some frames while the mobility of A is notified if the design requires it.

The key in inter-L2 mobility with Torii-HLMAC is that only edge switches need to update information in relation with the host change, and only in some cases, and this can be requested by a simple ARP message.

E. *Generalization of Torii HLMAC to any data center topology*

This paper has shown how Torii-HLMAC would work in a topology like the one described in PortLand. This topology is based on using commodity switches with all links of equal capacity and, although named as FatTree is better described as a Clos Network [8]. The difference between both concepts is that a Fat Tree [9] increases the capacity of its links the closer it is to the core switches, while in a Clos Network all links have the same capacity. Fat trees are also very suitable for Ethernet networks thanks to the aggregation compatibility with the Ethernet family growth (using links of 100Mbps, 1Gbps, 10Gbps, etc). Therefore, in practice, the use of one or the other will depend on the most desirable feature: less cost using cheap off-the-self components (Clos Network) or less wiring complexity (Fat Tree).

In the case of the Torii protocol, both (Clos networks and Fat Trees) could be used. In fact, Fat Trees are specially convenient for simpler wiring once there are four core switches, since four core switches provide the network with enough multipath forwarding and, in case of a link failure, still three alternative paths to be used. The only condition for using Torii in the different topologies is that nodes are organized into 'pods' so that from a core node (at the top) to its corresponding edge switches (at the bottom), links form a tree.

In Fig. 8, two different types of topologies with four levels of switches (instead of the three levels in the previous examples), in which the Torii protocol could be applied, are shown. The one on the left, a Clos network, would need more wiring, but it allows the use of 4-port commodity switches, and each host could apply up to 8 Torii-HLMACs addresses (since there are 8 core switches at the top). While the one on the right, a Fat Tree network, would need less wiring and the wires capacity increases towards the top (see the fatter links), but commodity switches could not be used, and each host could apply only up to 2 Torii-HLMACs addresses (only 2 core switches at the top), which could be enough in some circumstances. Choosing one or the other will always depend on the design requirements.

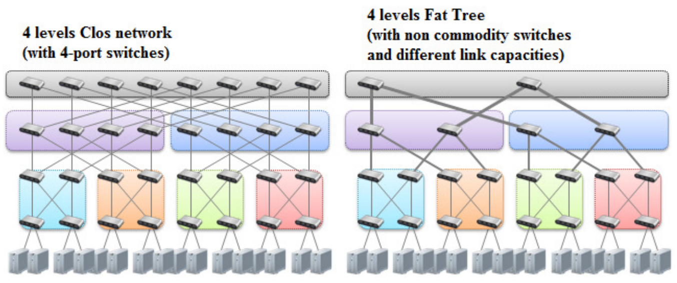

Figure 8: A Clos network and its equivalent Fat Tree network of 4 levels of hierarchy and 8 edge switches

Another design option could be the PortLand-alike topologies, the so-called "Fat Trees" (which in fact are Clos networks), the specific features of these topologies described in [1], as well as its scaling properties, make it particularly appealing for the data center. In general, a "Fat Tree" built from identical *k*-port switches can support 100 percent throughput among $k^3/4$ servers using $k^2/4$ switching elements. The topology should be organized into *k* pods, each connecting $k^2/4$ end hosts. Edge switches in each pod provide connectivity to end hosts and connect to aggregation switches in the hierarchy's second level. Core switches form the root of the fat tree, facilitating interpod communication.

The topology shown in the Torii-HLMAC description figures considered *k* equal to four. Torii-HLMAC could be used in fat trees with *k* up to 16. This is because we have 6 bits for the prefix (the first level of six in the HLMAC), so the number of switching elements $k^2/4$ should be smaller than 2^6 which is the number the HLMAC would cover.

$$k^2/4 <= 2^6 \rightarrow k^2 <= 64*4 = 256 \rightarrow k <= 16$$

The wiring complexity of a fat tree with 64 switching elements is high enough to even consider a bigger *k* topology. For this specific case of *k* equal to 16, which would mean *k* pods of $k^2/4$ end hosts, 16 pods of 64 end hosts, i.e. a total of 1024 hosts and the total bisectional capacity would be 1Tbps if links were all of speed 1Gbps.

IV. Conclusions

Torii-HLMAC improves PortLand in several ways: multiple addresses are automatically assigned in a distributed form without duplicates, instead of by a centralized module. Routing and path repair is distributed and performed based solely on the destination tree-based HLMAC address used, without routing tables at bridges, allowing high speed forwarding. In case of a link failure in a path, the bridge instantly selects an alternative path to reach the destination host and also notifies both edge switches serving origin and destination so that the non valid path is not chosen again, for a while. The multiple addressing allows load balancing based on a hash function, which can be designed specifically for different topology requirements and traffic models without changing Torii's main logic. The topology scales up to 6 levels plus roots and more, if needed. Torii's topologies are more flexible than PortLand's and a true fat tree can be set up, in which the links connected to the core switches are fatter (higher speed) in order to simplify the wiring instead. The independence of Torii-HLMAC from IP is clear as well.

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] A. Vahdat et al. Scale Out Networking in the Data Center. IEEE Micro, July/August 2010

[2] R. Mysore et al. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In ACM SIGCOMM, August 2009.

[3] M. Al-Fares, A. Loukissas, A. Vahdat. A Scalable, Commodity Data Center Network Architecture. SIGCOMM 2008.

[4] D. Kim, M. Caesar, J. Rexford. Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises, in Proc. ACM SIGCOMM, August 2008.

[5] G. Ibáñez, A. García-Martínez, J. A. Carral, J. M. Arco, A. Azcorra. Evaluation of Tree-based routing Ethernet. IEEE Communication Letters, IEEE June 2009 Vol. 13 No 6 pp. 444 – 446. DOI: 10.1109/ LCOMM.2009.090469

[6] G. Ibáñez et al. HURP/HURBA: Zero-configuration hierarchical Up/Down routing and bridging architecture for Ethernet backbones and campus networks. Computer Networks. Vol. 54, Issue 1, 15 January 2010, pp 41-56. http://dx.doi.org/10.1016/j.comnet.2009.08.00

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

[8] C. Clos. "A study of non-blocking switching networks". Bell System Technical Journal 32 (2): 406–424. 1953.ISSN 00058580

[9] Charles E. Leiserson Fat-trees: universal networks for hardware-efficient supercomputing, IEEE Transactions on Computers, Vol. 34 , no. 10, Oct. 1985, pp. 892-901.