TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett

RESEARCH ARTICLE

Torii: Multipath Distributed Ethernet Fabric Protocol for Data Centers with Zero-Loss Path Repair

Elisa Rojas[1], Guillermo Ibanez[1], Jose Manuel Gimenez-Guzman[1*], Diego Rivera[1] and Arturo Azcorra[2]

[1] Departamento de Automatica, University of Alcala, Alcalá de Henares, Spain
[2] IMDEA Networks, Leganés, Spain

ABSTRACT

This paper describes and evaluates Torii, a layer-two data center network fabric protocol. The main features of Torii are being fully distributed, scalable, fault-tolerant and with automatic setup. Torii is based on multiple, tree-based, topological MAC addresses that are used for table-free forwarding over multiple equal-cost paths, and it is capable of rerouting frames around failed links on the fly without needing a central fabric manager for any function. To the best of our knowledge, it is the first protocol that does not require the exchange of periodic messages to work under normal conditions and to recover from link failures, as Torii exchanges messages just once. Moreover, another important characteristic of Torii is that it is compatible with a wide range of data center topologies. Simulation results show an excellent distribution of traffic load and latencies, similar to shortest path protocols.

***Keywords*—** Ethernet; tree-based routing; routing bridges; data center; Shortest Path Bridges

***Correspondence**
J.M. Gimenez-Guzman, Departamento de Automatica, University of Alcala, 28805 Alcala de Henares, Madrid, Spain
E-mail: josem.gimenez@uah.es

1. INTRODUCTION

Data centers are nowadays one of the pillars of the Internet, whether or not consciously, being them used by most of users of the information systems. Data center networks are increasingly relying on Ethernet and flat layer two networks due to its excellent price/performance ratio and configuration convenience. But Ethernet layer-two networks do not scale. The main limitations for scalability are the flat address structure of Ethernet MAC addresses and the need of blocking active links to prevent broadcast frame loops. The use at data centers of known and regular topologies like fat trees has provoked the appearance of protocols specifically designed for these topologies that take advantage of the known network topology to implement layer-two routing and forwarding without the limitations of IP and layer-two protocols. Among them, protocols that use a hierarchical and topologically significant address structure that permits straightforward routing, are becoming more and more important due to its simplicity, performance and excellent scalability. PortLand [1] and, more recently, Torii, first proposed in [2], exemplify centralized and distributed versions of this approach. The majority of these proposals take advantage of topologies constituted by multiple complementary trees [3] that allow load balancing and multipathing [4]. However, just a few proposals benefit from creating multiple paths between different hosts.

Prepared using ettauth.cls [Version: 2012/06/19 v2.10]

PortLand is an outstanding architecture proposal for scalable data centers focused on the scale out [5] model and on topologically significant addresses. It uses the so-called FatTree (which is in fact a folded Clos) network topology based on interconnection of equal size pods of scalable size as the basic network component. PortLand uses a centralized control (fabric manager) and location-based pseudo MAC addresses. Addresses are assigned by a location discovery protocol executed at the switches and Up/Down [6] turn-prohibition is enforced to prevent frame loops. Paths are computed and routes installed at switches by the central element; in case of link failure, the central element installs the new routes at switches. However, this centralized route computation and installation at switches limits network scalability and reduces reliability. In contrast, our objective is to define a distributed, lightweight protocol for data centers that does not need forwarding tables, tolerates link failures gracefully and that is able to distribute traffic load evenly with no need of any centralized server.

In order to address the main challenges for current data center architectures, we briefly list and define the most important ones below:

- **Scalability**: Data center architectures should be easy to build and configure, e.g., wiring should not be too complex. The forwarding state should be also as low as possible since data centers usually interconnect thousands of final hosts. Finally, data centers should have the logic as distributed as possible in order to avoid congestion on a central manager and increase reliability.
- **Flexibility**: The topologies used in data center architectures should be as flexible as possible and not only restricted to one single configuration, so that administrators can easily expand and adapt their data centers for future demands.
- **Fast repair**: Data centers require high availability and therefore link or node failures should be tackled as soon as possible.

In this paper we define and describe Torii-HLMAC (from now on, Torii), a fully distributed protocol that makes forwarding in fat trees and other hierarchical data center topologies, simpler and more scalable. The protocol improves PortLand with simpler and fully distributed mechanisms applied to the same topology, but Torii is also extensible to real fat trees [7]. This protocol uses multiple simultaneous topological, tree-based, pseudo MAC addresses, inspired in TRE [8], to provide simple multipath forwarding, direct frame routing without tables and on-the-fly alternative path selection after link failure. These multiple addresses encode topological information of any port, making possible simple, hash-based, multiple path routing and load balancing without forwarding tables. Torii uses NAT of MACs at edge bridges to replace universal addresses by local (pseudo MAC) addresses.

The main contributions of this paper are: a) Torii's addresses assignment and address-based (tableless) forwarding; b) a performance evaluation of Torii against shortest path routing protocols regarding load distribution and latencies; c) a comparison with other data centers proposals and the contributions of Torii. The rest of the paper is structured as follows. In Section 2, the automatic addresses assignment mechanism is explained, while in Section 3 we describe the broadcast/unicast forwarding and path repair. In Section 4 we consider implementation issues and scalability of Torii and in Section 5 we perform the evaluation of Torii and compare it with shortest path routing protocols. Section 6 is devoted to the related work in the field and, finally, we recapitulate and conclude the paper with Section 7.

2. AUTOMATIC ADDRESSES ASSIGNMENT

Opposite to PortLand [1] or VL2 [9], which use a single topological address per host, the Torii protocol assigns multiple topological tree-based Hierarchical Local MAC (HLMAC) addresses. The key point is that each topological address precisely codes an alternative path to reach the host, making simple both multiple path routing and also rerouting of packets upon link failure via alternate paths on the fly.

For the sake of simplicity, we use PortLand's topology (Figure 1) to describe the address assignment, as it represents a typical hierarchical data center network with three levels (core, aggregation and edge) and divided into four pods. Thus, for the topology under study, first (upper) layer bridges are assigned just one HLMAC, second layer bridges get two HLMAC addresses (one per link to upper layer bridges) and so on in powers of two. HLMAC

Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett
Prepared using ettauth.cls

addresses are local (private) MAC addresses, so their U/L bit (universal or globally unique/locally administered) is set to 1. The 46 bits available for addressing purposes (after removing the U/L and multicast bits), encode by default up to 6 different hierarchical levels, with 6 bits for the first level and 8 bits for every other level. The HLMAC address 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.

Each node gets one or more topological tree addresses, existing a correspondence between the number of alternative HLMAC addresses and the number of core switches. There will be at least as many alternative HLMAC addresses at the edge switches as core switches in the topology, i.e. as many different paths as core switches. This is because every HLMAC prefix allows forwarding of frames over a tree rooted at a different core switch, then the HLMAC prefixes can be used to distribute traffic, on a hash base, among all available core switches. This address assignment method not only provides multiple alternative paths between pairs of hosts but also, at the same time, the multiple HLMAC addresses assigned to a host are directly inferable from each other just by changing the core prefix, simplifying multiple path routing and path repair on link failure. For example, for the first host on the left of Figure 1, which gets assigned 1.1.1.1., 2.1.1.1., 3.1.1.1. and 4.1.1.1. as HLMAC addresses, the only change is the core prefix 1, 2, 3 and 4 respectively. Note that all HLMAC addresses have the standard MAC address size (6 bytes), but for the sake of simplicity in notation, the bytes at the end, filled with zeroes, are omitted.

After the initial assignment, there will be no changes even when node or link failures occur, since Torii knows the complete topology and it is able to circumvent failed links or nodes. Therefore, this address assignment is done just right after the system is started up and this procedure is not necessary to be repeated unless there is an update in the network, as for example, when new switches are added.

In order to assign these addresses, the Torii protocol uses an extension of the Rapid Spanning Tree Protocol (RSTP), as defined in HURP [10], to build a spanning tree and to assign those hierarchical addresses to the bridges. Once the root bridge is connected on top of the core bridges (a function that may be implemented as a couple of root bridges for reliability reasons), which gets 0 as HLMAC address, the process of building the spanning tree from the root to the leaves starts and core bridges get assigned the addresses 1, 2, 3, 4. This iterative procedure of address assignment consists of Bridge Protocol Data Units (BPDUs) being sent by the parent bridge. Each BPDU contains the bridge HLMAC address (i.e. the HLMAC address of its root port) and the number of the designated port transmitting the BPDU.

3. FORWARDING AND ROUTING

Frame routing is directly performed by decoding the destination address, i. e. no forwarding tables are installed in any of the switches of the network. Once the HLMAC addresses are set, Torii switches need to distinguish between broadcast/multicast and unicast frames, and they also need to identify the direction of the frame: ”going up” or ”going down”, which is obvious by looking at the frame input port (lower side and upper side respectively). Once those two parameters are known, the logic applied at each switch of the topology is shown in the algorithm defined in Figure 2.

3.1. Broadcast Forwarding

First of all and looking at the pseudo-code, when a host A sends a broadcast frame, the switch serving that host chooses a prefix according to the result of applying a hash function to the source address and destination address fields of the frame header (to prevent disordering of frames belonging to the same flow). That 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) is kept unchanged, the frame source address A is translated (NAT of MACs) into the corresponding hierarchical HLMAC address (see Figure 3, in which A source address is translated into 1.1.1.1. 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 selected and the address translation from global MAC to local HLMAC addresses is done, the frame is directed up to the matching core switch and then replicated down to every link except the one associated to the input port as shown in Figure 3. Since only one

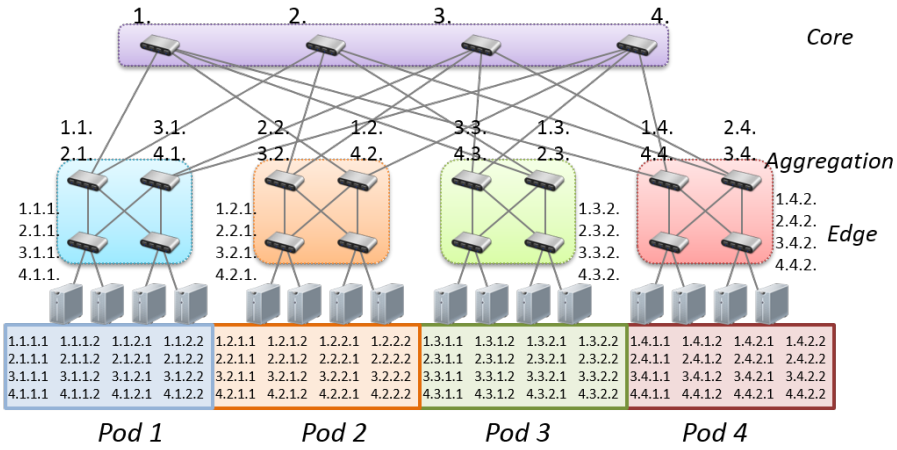


Figure 1. Multiple hierarchical addresses (HLMAC) assignment for Torii.

Forwarding and Routing Algorithm

```
1:  if frame is BROADCAST or MULTICAST then
2:      if frame goes UP then
3:          if switch is edge then replace source host MAC address by HLMAC address
4:          Forward frame through the HLMAC port (see Note I)
5:      else if frame goes DOWN then
6:          if switch is edge then replace destination HLMAC address by host MAC address
7:      Broadcast frame downwards (see Note II)
8:  else if frame is UNICAST then
9:      if frame goes UP then
10:         if switch is edge then replace source host MAC address by HLMAC address
11:         Forward frame through the HLMAC port (see Note III)
12:     else if frame goes DOWN then
13:         if switch is edge then replace destination HLMAC address by host MAC address
14:         Forward frame through the HLMAC port (see Note I)
```

Note I: Forwards through the next port according to the HLMAC address (up port if the frame comes from a down one or vice versa).
Note II: Broadcast only through the ports located down in the hierarchy except through the input port.
Note III: Same as Note I, 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).

Figure 2. Forwarding and routing in Torii.

core switch is used, there are no down-up turns and the communication remains loop-free.

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

Torii protocol does not include special features to process multicast frames, so the same forwarding mechanism explained for broadcast frames is also applied to multicast frames. As it can be seen, broadcast forwarding is performed across the spanning tree as it occurs in classical Ethernet. However, the difference between Torii and the Spanning Tree Protocol is the possibility of choosing one of the multiple (four in the figure) trees available to distribute the traffic. To improve

Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett
Prepared using ettauth.cls

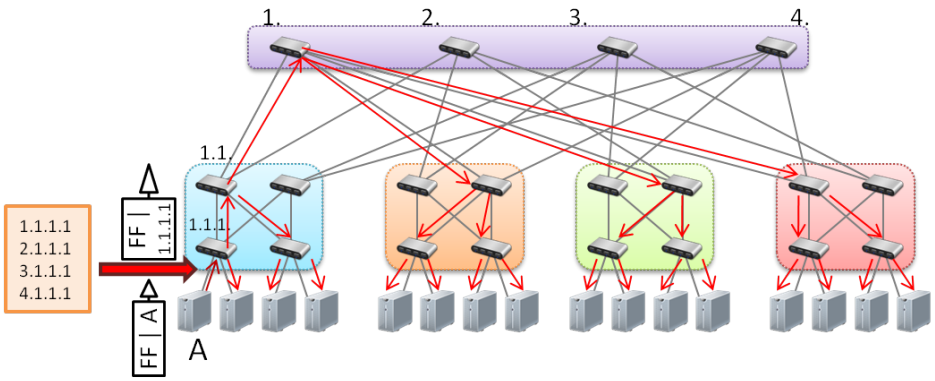


Figure 3. Broadcast frame from host A. The broadcast address remains the same while the A source address is translated into $1.1.1.1$ at edge bridge in the frame when prefix 1 has been chosen by hash.

scalability, an ARP proxy function may be implemented distributed at edge bridges [11], learning from all ARP Request and ARP Reply frames, or centralized, as in PortLand.

ARP messages in the Torii protocol are broadcast like standard ARP messages, being the only difference in the fact that the ARP Request broadcast packet in the up direction is forwarded only via one link in its upward path towards a single core switch; standard broadcast with flooding is performed for downwards forwarding. Flooding in Torii is thus enhanced compared with the standard ARP broadcast procedure. For that reason, we can consider that the impact of ARP in Torii is always lower than the one we have in standard layer-two protocols. In [12], authors study the scalability of the ARP protocol, assuming that the amount of ARP traffic scales linearly with the number of hosts. For example, for a network comprised by 25000 hosts, from [12] we can derive that we would expect 11706 ARPs per second or 5.9Mbps of ARP traffic to arrive at each host at peak. As data center links are currently 1Gbps and are evolving towards 10Gbps technologies, we can consider this traffic negligible. The cost of processing ARP messages at switches is negligible, but for end hosts can be quite high because every host must process every ARP Request messages received, so the use of ARP proxies is recommended at edge switches in big data centers in order to minimize the impact on host performance.

3.2. Unicast Forwarding

In the case of unicast frames, a hash function is also applied to select the prefix (i.e. the core bridge). Unicast communications can be bidirectional (paths are congruent or symmetric, same path used in both directions) or not, both cases are acceptable for Torii. In any case, the forwarding path is always determined by the destination address and its core prefix.

Once the unicast frame arrives at the source edge switch, this switch translates both addresses (NAT of MACs). The origin address is translated into the corresponding 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 address is always known by a previous ARP Request/Reply message, which will be always sent before any unicast frame). For instance, in Figure 4, prefix 1 was chosen and, because of this, the origin B is translated into 1.3.1.2., while the destination A is translated into 1.1.1.1., which is known by the previous ARP Request message (see Figure 3).

If the learnt HLMAC address at the edge switch had a different prefix when the ARP Request/Reply message was received, the corresponding HLMAC address would be easily deduced. For example, if prefix 2 had been chosen instead of 1 (for load balancing purposes), B would be 2.3.1.2. and A address could be deduced as 2.1.1.1. from the previously learnt A HLMAC address 1.1.1.1. (conveyed in the previous ARP Request message as in Figure 3), just by changing the core prefix, since only the prefix part of the HLMAC address differs

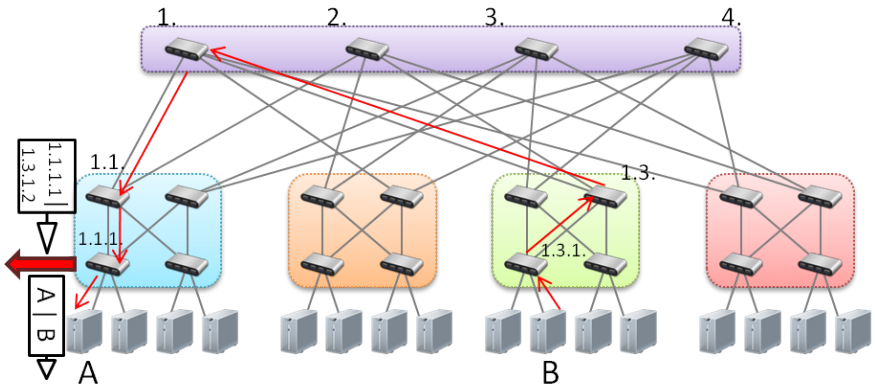


Figure 4. Unicast frame from B to A. Both addresses (A and B) are translated at the edge switches, which already know them from the previous ARP messages. In this case A is translated into $1.1.1.1$ and B into $1.3.1.2$.

among the HLMAC addresses. This is an advantage of Torii's address assignment, which makes all HLMAC addresses completely deducible from any other once a single HLMAC address is already known.

After the prefix is selected and the MAC address translation is done, the frame is forwarded up or down according to the destination HLMAC address. This is done by checking the current switch HLMAC address and the destination HLMAC address. 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. At switch 1.3., the frame with destination 1.1.1.1. is known to be located in another pod (since 1.1.1.1. does not contain the prefix 1.3.), and that switch is aware that the frame needs to be sent to the port connected to switch 1.; then switch 1. will send the frame through its port 1 to reach switch 1.1. and so on.

Finally, at the destination edge switch, if the frame is an ARP Reply message, the two-address tuple (the destination MAC address and the translated HLMAC address) is saved in the address translation table. In this way, due to the ARP Request and Reply messages, necessarily exchanged before any communication, origin and destination edge switches will be capable of translating the addresses of successive unicast frames received.

In this example, notice that unicast frames will always use the same prefix in both source and destination HLMAC addresses. Therefore, unicast frames having source and destination HLMAC addresses with different prefixes could be used for special purposes such as notifying an action (for example, a failed link when found) to any switch in the topology. Nevertheless, a free bit in the HLMAC address could also be reserved for these special events (see Section 4.2 for further details).

3.3. Path Diversion

We describe now path diversion, which is the mechanism applied to frames when their path to destination is broken by link or bridge failure.

When a link fails, no messages are exchanged and the assigned HLMAC addresses remain exactly the same, only the switches connected to that link will know that the link is down and no longer usable to forward frames. Therefore, when a frame arrives at a switch and its destination HLMAC address indicates that it should be sent through the port attached to the failed link, the path diversion procedure starts.

Torii's path diversion mechanism consists of assigning a different path to the frame on the fly, once the frame arrives at the point where the current route is broken. All the switches in the network, but the core ones, have two or more HLMAC addresses, which means that all those switches have two or more different routes through two or more different core switches. Thus, switches can decide unilaterally a different route for the frame just by assigning it a different HLMAC address, which can be directly deduced on the fly from its current HLMAC address just by using a different prefix (because an important feature of Torii is that all assigned addresses just vary in the prefix). Once the new HLMAC address is decided, indirectly the

DOI: 10.1002/ett
Prepared using ettauth.cls

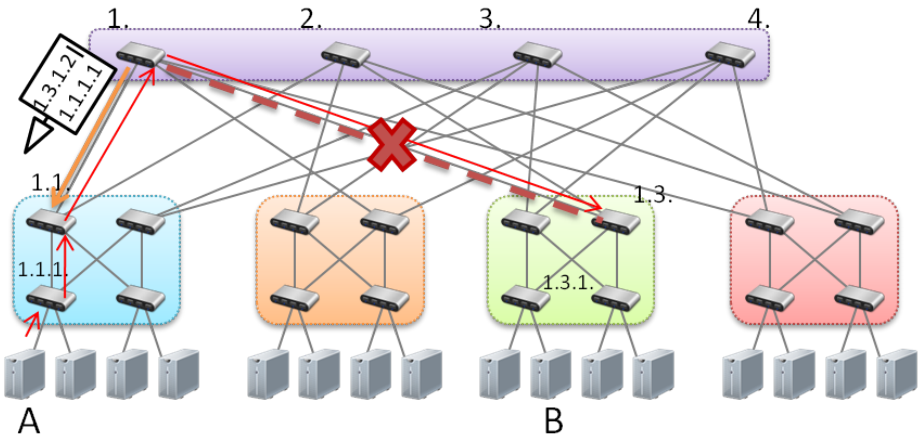


Figure 5. Unicast frame from A to B. Link from switch 1. to 1.3. is down, when the frame arrives at 1. it is forwarded back to 1.1. until it reaches its new closest alternative path, which is the one with prefix 2.

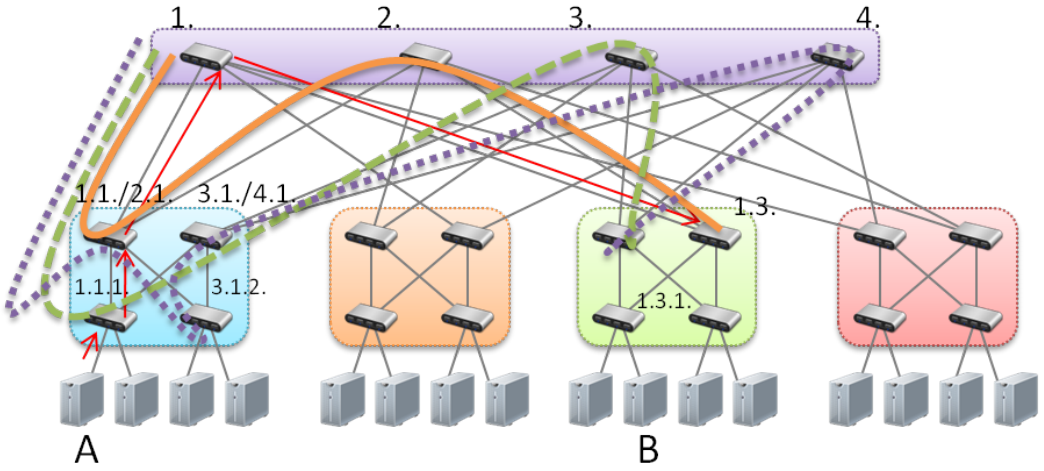


Figure 6. Example of the path diversion procedure with multiple link failures.

path is changed, without needing to change any table nor previous learning, neither sending any notification to any other switch.

As there is no need to follow a specific new path, the first and closest alternative path available is selected. By closest, we mean the alternative path that requires fewer steps back, sometimes none, to continue the forwarding. Therefore, the frame is forwarded back (if necessary) until it reaches the new path and it is then forwarded by using the new path as shown in Figure 5, in which the frame is forwarded back from 1. to 1.1., where it founds a path to reroute the frame through core switch 2. (from 2. it will later go to 1.3. and 1.3.1., or 2.3. and 2.3.1., and reach destination as usual).

In order to avoid loops or any other problem while rerouting traffic, diverted frames are forwarded back step by step and always in the same order. The reason why frames are forwarded back instead of to any other switch is that previously used switches have much less probability of having failed than any other random switch still to be selected. In Figure 6, we show the steps followed by the frame that found a failed link as in Figure 5 (link 1. $\leftrightarrow$ 1.3. is down). First of all, it goes back to switch 1.1. to bounce later to core switch 2., but if link 1.1. $\leftrightarrow$ 2. is down, it goes back again until switch 1.1.1.. Once in switch 1.1.1., it tries to go to switch 3.1., but if link 1.1.1. $\leftrightarrow$ 3.1. is also down it needs to use the remaining core switch 4., for which it needs to do a Z *reroute*, which means that

the frame will go from 1.1.1. to 1.1. to 3.1.2. to 3.1. and finally to 4.. While forwarding back, there is no need to change anything in the frame and switches know the frame is being forwarded back because they received it in the port through which it should be sent. However, the *Z reroute* needs to be indicated since otherwise the switch receiving it will consider the frame as a normal one. Therefore, the prefix of the source HLMAC address is temporarily set to 0 which forces the receiving switch (1.1.) to send the frame down again instead of up (send it to 3.1.2. instead of to 1.), and later is set to the prefix of the new path (in this case 4). Finally, if the path through core switch 4. is also broken (which means 4 of 20 links in the topology failed, i.e. 20% of failed links), the frame is discarded since we consider there are too many failures in the network and diverting traffic stops being functional.

3.3.1. Notification messages

It is important to notify the edge switches to prevent the use of the failed path and start using a new one. The path diversion mechanism just redirects temporarily frames through an alternative path so that they are not lost, while edge switches need to be notified in order to stop the effort of path diversion, but they do not necessarily have to choose the same path that was used by the path diversion mechanism. In this manner, the path diversion mechanism acts locally for lossless communication, while edge switches can make more global decisions for the new paths taking into consideration other affected flows from directly connected hosts that traversed the currently failed path.

Notification messages are only exchanged in case of failure, so its impact on network performance is expected to be negligible. There are two types of notification messages: one for the destination edge switch (which is the same frame being diverted) and one for the source edge switch (which is a copy of the frame being diverted). In the case of unicast frames, since the edge switches need to be notified to avoid the selection of the failed path, the source HLMAC address will be translated into the new HLMAC address, while the destination address will remain unchanged. In this way, the redirected frame serves also as a notification of the link failure to destination as well, because its HLMAC prefixes will not coincide, then it will be considered a special frame; if this frame is forwarded back to source it will also indicate the failure to the source. Alternatively, this notification could also be done by using a dedicated bit of the HLMAC address instead of the prefix (Section 4.2).

The notification indicates to both switches, the one serving the destination host and the one serving the source host, not to use the failed path (shown in the destination prefix) and start using a new one instead. For instance, in Figure 7, the source is translated into 2.1.1.1., while the destination remains as 1.3.1.2.. However, since it is processed as a failure notification frame, it is sent to 2.3.1.2. through core switch 2., i.e. the frame does not follow the path of destination for routing, but conversely, the destination with the prefix of the source. The frame still contains the information to be delivered and at the same time the frame indicates that the path through core switch 1. failed to the destination edge switch. The same happens to the frame forwarded back to source, which is known to be only a notification because the prefixes are different and is known to be forwarded back because it is received at the port through which it should be sent to destination. After both notifications are received, both edge switches know about the failed link and will not assign again that path after hashing, but an alternative one instead.

This path-failure notification lasts for a configurable timer. Once the timer expires, frames will start using the old path again, thus activating the path diversion mechanism again if the failure still persists.

It is important to notice that there is only one extra message forwarded back to notify the source when a path fails, and it is optional. This is because the rerouted frame to destination is a notification itself, while notifying source might not be necessary specially when there is traffic in the other direction (if so, the frames in the opposite direction will notify our source edge switch). Therefore, if no notification to the source edge switch is configured, then the overhead in the network is null. However, if some traffic is purely unidirectional in the network, we might configure notifications to source edge switches (otherwise frames will be continuously diverted, but losslessly rerouted), in which case in the whole network there would be one single message per failed path and active source edge switch ($N_{fail} \cdot N_{edge}$ messages, where N_{fail} is the number of concurrent failed paths and N_{edge} the number of edge switches in the network with active flows originating on them), e.g., in the case of the single failure shown in Figure 7 and considering all edge

Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett
Prepared using ettauth.cls

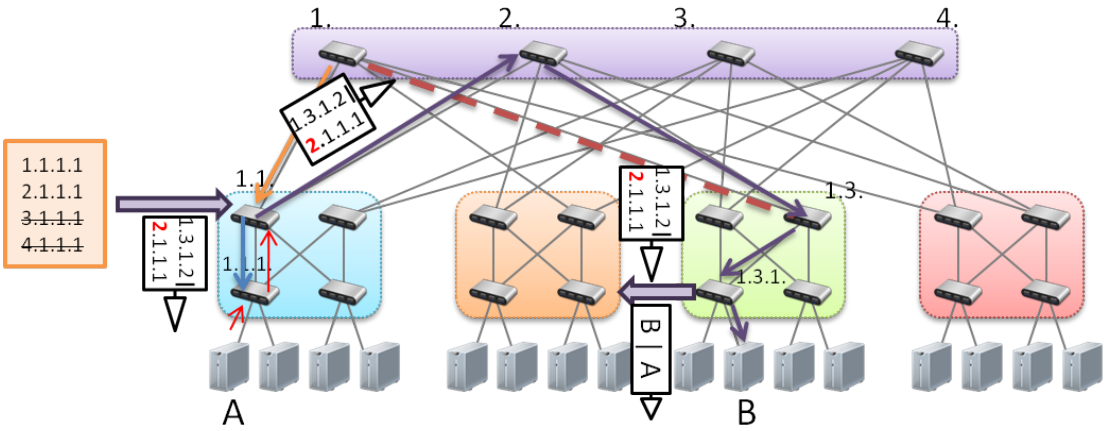


Figure 7. Unicast frame from A to B. Once the frame is back to $1.1.$, the source HLMAC address is translated into the new path HLMAC address (from $1.1.1.1.$ to $2.1.1.1.$) and sent to core switch $2.$. The destination HLMAC address remains the same, $1.3.1.2.$, but the frame is interpreted as a redirection notification frame so it is sent towards the new HLMAC address: $2.3.1.2.$. It will also be sent to destination and optionally back to source to notify both edge switches of the failed path.

switches are emitting traffic at that moment, the number of messages would be 8. Therefore, when path repair is needed, not only alternative paths are decided on the fly, but there is no significant overhead in the network.

4. IMPLEMENTATION OPTIONS AND SCALABILITY

4.1. Use of Virtual Machines at Hosts

In current data centers, multiple virtual machines (VM) are active at the same time on each physical host. IEEE 802.1Qbg (Edge Virtual Bridging) [13] standard group was created to simplify management of virtualized machines and its virtualized interconnections. The approach is to interconnect all virtual network terminations (one per VM) via the edge port of the physical bridge, instead of interconnecting them internally, in order to facilitate VM visibility to the physical bridge. The Reflective Relay function implements VM intercommunication similar to a bridge: unicast frame forwarding between two VMs, broadcast forwarding to all other VMs of the host and replication of unknown unicast frames. The VSI Discovery Protocol (VDP) implements mechanisms to inform the physical bridge about the creation/deletion of a VM. The commands are: Preassociate (to inform the switch of the VSI and port-profile of an intention to associate), Associate (to execute the previously requested preassociation) and Deassociate.

4.1.1. Address Assigment to Virtual Machines at a Host

Every VM gets assigned a specific MAC address by the virtualization kernel. In the most generic case, Torii would assign only one HLMAC address per physical host, but there is room in HLMAC address range to assign one HLMAC per VM. In practice, Torii only uses the first four bytes of HLMAC addresses, so the last two bytes can be used to distinguish among VMs of the same host by assigning them one at a time when the VM sends its first ARP message. Therefore, every host could have up to $2^{16} - 1 = 65535$ VM working at the same time (since number 0 is not used for HLMAC addresses). Eight bits, allowing up to 256 VM addresses per host, would suffice for years to come.

4.2. Other HLMAC Address Assignment Alternatives

In this proposal, Torii takes 1 byte of the 6 of the HLMAC address per hierarchical level, which means 4 bytes and the remaining 2 bytes could be used for specific per-VM addressing. Nevertheless, if more hierarchical levels were 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

protocol. Moreover, free bits could be used as flags to indicate whether a frame is a notification or not as already mentioned instead of other combinations of bytes.

4.3. Layer Two Mobility

Virtual machine mobility is commonplace in data centers and used for multiple purposes like increasing server utilization, redundancy and replication, server migration and others. Regarding layer two mobility, when a host (or virtual machine in a host) A communicating with another B, moves from one edge switch to a different one, frames follow the next procedure:

(i) If the frame goes from A to B:

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

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

(ii) If the frame goes from B to A:

Several options are possible with different costs regarding broadcast messages, depending on the requirements. In this case (frame from B to A), 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 address and send a special message (ARP Reply) towards B with the new A's HLMAC address. If 802.Qbg EVB is used, the Associate message can be used by Torii edge bridges to issue a gratuitous ARP message to inform all the network of the new HLMAC address of a VM being connected. Previous edge bridge of host A takes note of the new HLMAC of the migrated VM and may forward received frames to its new destination.

Both cases require broadcasting of frames in order to support lossless frame deviation. However, alternatively, broadcasting could be reduced by discarding frames while the mobility of A is notified if the design requires it. The key aspect regarding layer-two host mobility with Torii is that only the edge switches need to update information in relation with the host change of point of attachment, and only in some cases, and this can be requested by a simple ARP message and without any kind of address manager.

4.4. Beyond the FatTree Topology

In the previous sections we have introduced Torii and explained its forwarding and path diversion mechanisms by using the well-known topology of PortLand, the so-called FatTree. However, one of the main characteristics of Torii is its applicability to many different data center topologies, thus being more flexible than those architectures that require specific topologies. Below we briefly compare the suitability of different data center topologies for Torii.

4.4.1. True fat tree topologies vs. PortLand (Clos)

The topology used in PortLand is based on the scale out model, i.e. it uses many commodity switches with equal capacity links and, although named as FatTree (in relation with the fat tree topology), it is better described as a Clos network [14]. The difference between both concepts is that a fat tree [7] increases the capacity of its links the closer they are to the core switches, while in a Clos network all links have the same capacity. True fat trees are very well suited for Ethernet data center networks thanks to the built-in aggregation capability of the Ethernet switches (capable of using and aggregating links of 100Mbps, 1Gbps, 10Gbps, etc). Therefore, in practice, the use of one or the other will depend on the most desirable feature: lower cost using cheap off-the-shelf components (Clos network) or much less wiring complexity (fat tree). It is worth noting that wiring costs and complexity escalate rapidly when all links have the same capacity because the number of links grows an order of magnitude. In fact, fat trees are particularly convenient for simpler wiring once there are 4 core switches, since 4 core switches provide the network with enough multipath forwarding and, in case of a link failure, there will be still three alternative paths to be used. Choosing one or the other will always depend on the design requirements.

Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett
Prepared using ettauth.cls

4.4.2. AB FatTree and Diamond

The AB FatTree (proposed in F10 [15]) and the Diamond [16] topologies are both variants of the PortLand FatTree. The former aims to improve the fault tolerance while the latter aims to shorten paths between final hosts. In both cases, core switches are defined as in PortLand: there are levels of hierarchy and no cross-links between same-level switches. Therefore Torii can be applicable in these topologies while exploiting their advantages in comparison to PortLand.

4.4.3. BCube and DCell

These topologies ([17] and [18], respectively) have the peculiarity of using final servers as routing devices as well as the switches. Torii can also be applicable to these topologies, but first we should define the core switches (i.e. the highest hierarchy switches), which would be every switch in every DCell/BCube group.

4.5. Scalability

Once we have explained the applicability of Torii to different topologies, in this section we address the scalability of Torii by comparing it directly with PortLand, which is already a quite scalable data center proposal. Scalability is an important issue in Ethernet networks, as described in [19]. For the comparison, we will use different hierarchical topologies which are applicable to Torii and, at the same time, define common examples of data center topologies of two, three and four levels of hierarchy. In Figure 8, different types of topologies are shown with different levels of hierarchy (2, 3 and 4). In each topology, the upper level represents the number of core switches while the lower level represents the edge switches. All of these topologies are applicable to Torii, but only the ones on the blue frame are applicable to PortLand; they are the so-called FatTree topology for $k = 4$ (up) and for $k = 6$ (down), being k the number of ports per switch.

4.5.1. Forwarding state per switch

In Torii, edge switches store a look up table to translate MAC to HLMAC for source and destination hosts and its length correspond to the number of active VMs, not the total number of VMs in the network. These translation tables work as ARP proxies at the same time.

Torii switches do not need forwarding tables at all because frames are routed just by decoding a part of the destination address at every stage. For every frame received, output port is obtained as a logic operation result, so no table is strictly required since switches in Torii only need to store the HLMAC addresses that were assigned to them, which are limited by the number of core switches (i.e. up to 9 entries in the worst case shown in the previous figure), independently of the active VMs. In PortLand, forwarding state depends on the number of ports of the switches. In particular, as stated in [1] "Required state for network connectivity is modest, growing with $k^3/2$ for a fully-configured fat tree built from k-port switches" (being k the number of ports of the switches).

Therefore, while Torii switches will need to store up to 10 entries in the worst cases (10 core switches are enough to allocate thousands of VMs for most data center networks), PortLand entries increase with $k^3/2$. Thus, while PortLand increases its size and its forwarding tables, Torii can reduce the number of link and switches by aggregating them and *fattening* the network (see Section 4.5.4).

4.5.2. Number of messages for routing and recovery

In Torii the only messages interchanged for routing are those performed upon initialization to assign the multiple HLMAC addresses. Upon link failure, there is no need to recompute and modify routing tables at switches, frames are rerouted automatically in a distributed way. Although Torii works in a distributed manner, it is also possible to have a centralized maintenance as an option. In PortLand, the Location Discovery Protocol (LDP) is used to discover the network topology and obtain the addresses. Upon link failure, it is reported to a centralized fabric manager that recomputes and modifies the routes at switches.

4.5.3. Number of broadcast messages

Distributed ARP proxy at every edge switch is possible in Torii and efficient as shown at [11]. ARP Request messages (standard or gratuitous) sent from hosts feed these edge proxies. Thus, the number of broadcast messages in the network will depend on the number of edge switches. A centralized proxy scheme as in PortLand is also possible.

In PortLand, broadcast of ARP Request messages is prevented with the use of a centralized ARP proxy. This proxy must serve the whole data center, serving

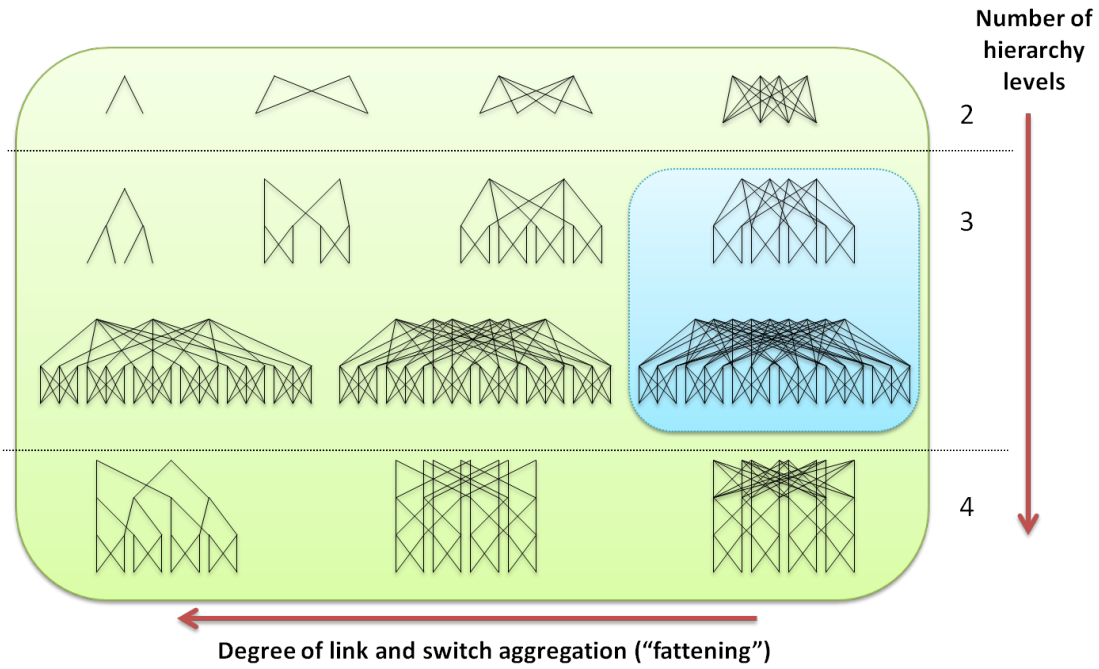


Figure 8. Examples of hierarchical data center topologies.

responses to all ARP Request messages. According to [11], typical ARP Request rates for a Yahoo data center are 0.028 requests/source/second on average and 0.25 on peak (although it also states that ARP traffic has a very high variability). This means on average 2800 requests/second and a peak of 25000 requests/second for a network with 100000 hosts, so the centralized ARP proxy could become overwhelmed.

4.5.4. Number of VMs and size of the network

The main limitation of PortLand is its strict requirement for the use of the so-called FatTree topology with an enormous number of equal capacity links. For the size mentioned at [1] of 100000 hosts and in order to support a throughput of 100%, the network would consist of 74 pods, each composed of 74 switches, and 1369 core switches. Besides the non-standard value for the number of switch ports, the cabling volume and complexity for such a network would be excessive [5].

Torii can aggregate links and reduce the number of ports and switches, so that the network does not grow as a function of powers of k (the number of ports of the switches), but it slightly changes while links become *fatter* (higher capacity) to support a higher throughput.

5. PERFORMANCE EVALUATION

A quantitative performance evaluation has been conducted by simulating the aforementioned three-level data center network (Figure 9) running Torii. We compare Torii with a basic shortest paths routing protocol (SP) using Djikstra algorithm [20] without Equal Cost Multiple Path routing. Our objective was to have a first confirmation that Torii performance was similar to SP and likely better as Torii uses hash-based multipath routing. More concretely, we have used OMNeT++ [21] discrete event simulator version 4.2.2 in conjunction with INET framework version 2.0.0 [22]. 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. In these simulations, we have used UDP traffic between different and random hosts in the data center topology. Note that we have not considered TCP traffic because its complex behavior (with mechanisms like slow start, flow control and congestion control) could mask the comparison between Torii and SP. The traffic model used is taken from [23, 24]. A flow generator (flow interarrival times are exponentially distributed with mean IAT, ranging from 0.1s and 1.6s) installs flows in the network by randomly selecting a

Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett
Prepared using ettauth.cls

pair of source and destination hosts. Each flow carries on average 34.8MB of data (following a truncated Pareto distribution between 8MB and 8GB) at 0.5Mbps (30% of flows), 1Mbps (60%) or 10Mbps (10%). With these throughput values, flow sizes and flow interarrival rates, we can compute the mean number of simultaneous active flows (N) used in the simulations using Little's Law, i.e. $N = T/IAT$, being T the mean flow duration, which is computed as the quotient between the mean flow size and the mean throughput. From the above-mentioned data, the mean number of simultaneous active flows ranges from 99.4 to 1590.9, which is a range large enough to evaluate the system under very different load conditions. Each simulation runs for 10000 seconds and packet size is 1500 bytes. We have chosen packet sizes of 1500 bytes as it is the most representative packet size due to the maximum transfer unit used in Ethernet networks. However, we have experimented with different packet sizes obtaining the same conclusions.

The performance evaluation and comparison of Torii vs. SP has been carried out in two steps. In the first one we study how the traffic load is distributed across the network and in the second one we study the average delay of packets.

5.1. Load Distribution

Load distribution capabilities for both SP and Torii have been studied by means of the link utilization. If we analyze the structure of the three-level data center network under study we can notice three different types of links: core, aggregation and edge links. While core links (CL) comprise all the links between a core bridge and a bridge from the aggregation level, aggregation links (AL) are those that connect aggregation and edge bridges. Finally, edge links (EL) are the links that connect edge bridges to hosts. In Figure 9 we show this link classification, marking CL in red, AL in blue and EL in green.

Due to the symmetry of the network, we can notice that an ideal load distribution would rely on having the same link utilization in all the CL and a similar reasoning holds for AL. Note that these conditions do not include that the utilization of CL must be the same to that of AL, i.e. the utilization of CL and AL can be different (and in fact it will be). Also note that we are not considering EL, as its behavior is the same independently of the use of SP or Torii, as the utilization of this part of the network is not affected by the routing mechanisms.

In Figures 10 and 11 we show link utilization for the proposed scenario with IAT=0.1s and for SP and Torii respectively. As it can be shown from both figures, EL utilization is not affected by the protocol used and, at a first sight, it seems that the traffic is distributed better along links in Torii than in SP, considering that the ideal case would be that link utilization would be the same for all the CL and the same behavior for the AL. To study this performance in more detail we have considered the mean and coefficient of variation (CV, the relation between the standard deviation and the mean value) of the link utilization for all the CL and, similarly for the AL. In Figure 12a we show the mean utilization for the CL and AL for both protocols and for different IAT. As it can be shown and could be expected, CL transport the same amount of data independently of using SP or Torii so the mean utilization is the same across all links. The same conclusion arises from the AL. For this reason, the study of the CV of the links that belong to the same group (CL or AL) will indicate us how well is load distributed along those links, being better distributed as the CV is lower. In Figure 12b we show that CV for different values of IAT, concluding that load is balanced much better in Torii than in SP as CV is $3 - 4.5$ times higher in SP than in Torii.

5.2. Delay

The next comparison between SP and Torii consists of the comparison between the average delay experienced by packets at every destination. In Table I we show the mean value, standard deviation and 95th percentile for the average delay obtained for different values of IAT and for SP and Torii. From that table we can conclude that Torii is able to slightly outdo SP in terms of average delay of packets. Intuitively, if we consider a relation between the shortest path and the minimum delay path, SP should behave very well in terms of latency, as it chooses the shortest path for all packets. In a first sight we can perceive that Torii is able to achieve those good delay values of SP. And moreover, it is able to slightly improve those values due to the fact that Torii is able to better distribute load in the network, so the queueing delay experienced in Torii is expected to improve that of SP.

In Figure 13 we show the cumulative density function of the average delay for both protocols and for the extreme

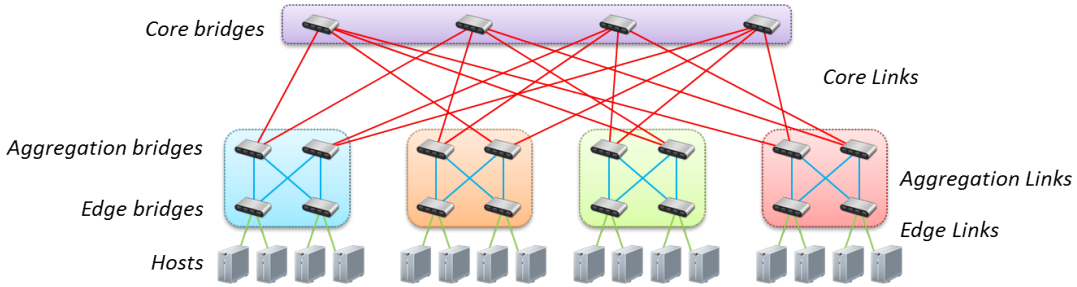


Figure 9. Link classification for load distribution.

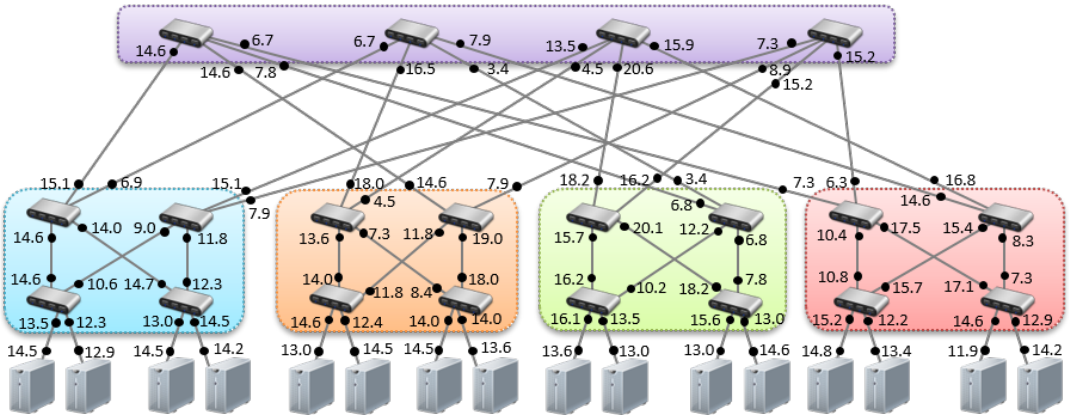


Figure 10. Load distribution for SP and IAT=0.1s.

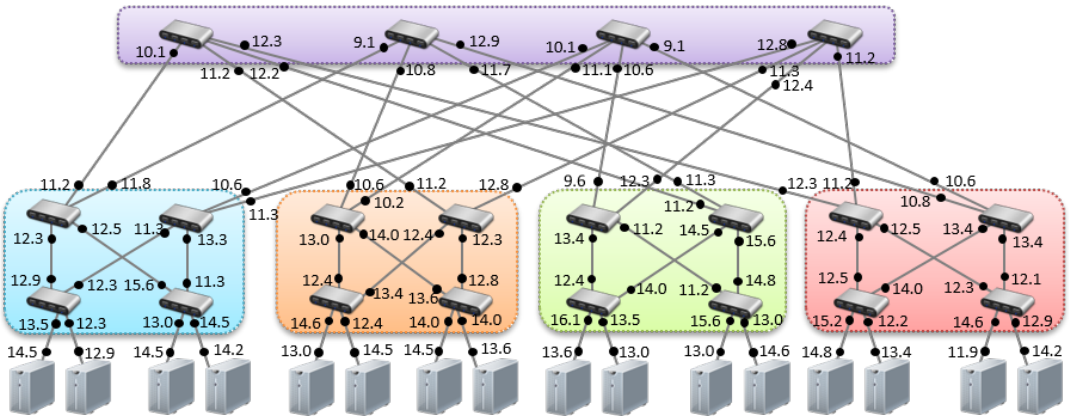
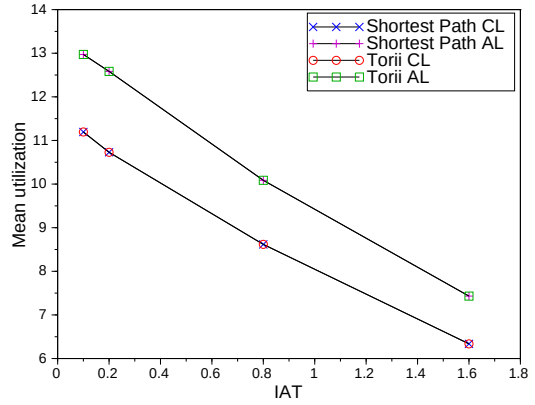


Figure 11. Load distribution for Torii and IAT=0.1s.

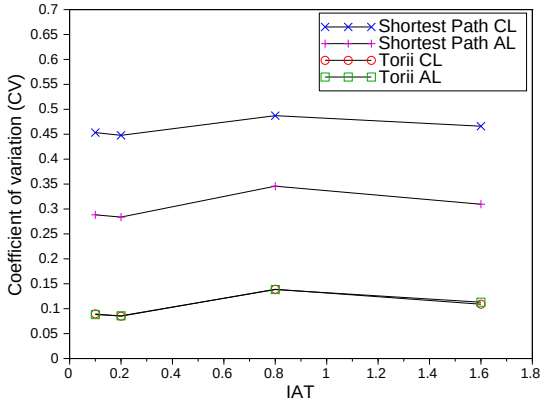
Table I. Average delay comparison (expressed in milliseconds).

	Shortest Path			Torii		
	Mean	Std dev	95th percentile	Mean	Std dev	95th percentile
IAT=0.1s	0.522	0.023	0.554	0.512	0.018	0.541
IAT=0.2s	0.507	0.024	0.565	0.506	0.026	0.548
IAT=0.8s	0.509	0.036	0.602	0.500	0.033	0.587
IAT=1.6s	0.503	0.029	0.569	0.499	0.029	0.566

Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett
Prepared using ettauth.cls

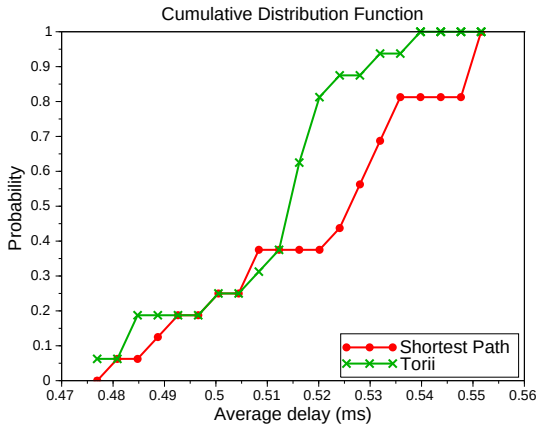


(a) Mean utilization.

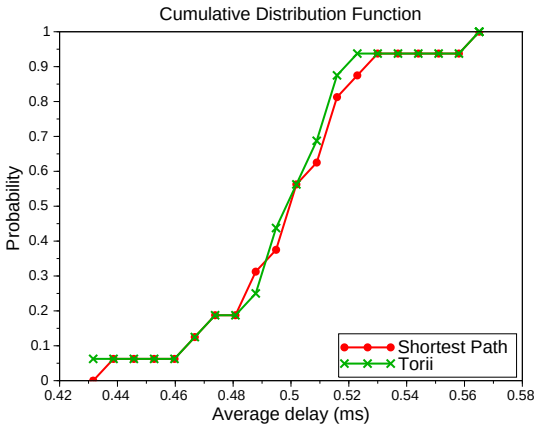


(b) Coefficient of variation.

Figure 12. Comparison of SP and Torii for different IAT (expressed in seconds).



(a) IAT=0.1s.



(b) IAT=1.6s.

Figure 13. Cumulative Distribution Function for the average delay.

IAT considered, 0.1s and 1.6s. In both cases we conclude that delay is similar, but in some cases it is better for Torii, even considering that the average delays for SP were expected to be very low.

6. RELATED WORK

Different approaches to implement a data center fabric have been recently proposed to overcome the limitations of Spanning Tree Protocol and the configuration complexity of Multiple Spanning Tree Protocol. Generic protocols like TRILL RBridges [25] and SEATTLE [26] are also applicable to arbitrary topologies, while Torii is applicable to hierarchical data center topologies with pods (Clos or fat tree networks) and PortLand is only applicable to the FatTree topology. TRILL and SEATTLE use link-state routing protocols in layer two while Torii and PortLand use topological address for routing. A comparison of some of the above protocols with other recent data center proposals, is shown in [1], where the advantages of PortLand become apparent, with the exception of the use of a central manager for address resolution path computation and path repair.

In the case of proposals of data center fabrics with specific network topologies which are previously known, PortLand [1] uses pseudo MAC addresses assigned by

a discovery protocol with hash-based load distribution and Up/Down turn prohibition to prevent loops, but it is severely limited by the central manager performance. VL2 [9], whose manager is distributed, uses Valiant Load Balancing (VLB) to distribute the traffic among the network with significant complexity due to encapsulation, it does not optimize forwarding state and the efficiency of VLB load balancing is severely reduced if per-flow load balance is performed to avoid frame disordering. DCell [18] and BCube [17] are data center architectures based on specific topologies alternative to the Clos network [14] and the so-called FatTree first proposed in [27], which offers the best performance according to [3]. However, these two proposals are only applicable to specific data center topologies, need specific routing protocols and probing, wiring is complex for DCell, which at the same time has a lower bisection bandwidth than fat tree topologies, and repair seems to be slow for both of them. Finally, Diamond [16] presents an improved fat tree network that shows better performance in terms of route paths length and edge to edge delay, but its routing protocol FAR needs some link-state information and two tables for forwarding, while Torii is also applicable to that topology and it is tableless.

In addition to the above, Torii is not only fully distributed (with no manager at all) and its forwarding state is very low (switches only store HLMAC addresses assigned at the beginning and no extra periodic messages are needed, nor any other type of link-state messages), but it has multiple paths between final hosts in order to efficiently balance the network's load and on-the-fly path repair. Furthermore, Torii is perfectly compatible with proposals related to automatic assignment of addresses, such as DAC [28] (which shows excellent performance when applied to BCube for example), GARDEN [29] (though being centralized is a clear disadvantage) and ALIAS [30]. These schemes can be enhanced with Torii to obtain tableless routing and instant repair, which are not possible with other routing protocols.

Minimizing reconfiguration time is crucial. Concerning repair times, convergence times for a rearrangement after a link failure is also crucial in data center networks, since a single point of failure might cause some resources to be unavailable during that time and that might not be acceptable at all in some circumstances. In this case, the Aspen Trees proposal [31] considers these times are fundamental, because the value of this convergence time for link-state routing protocols like OSPF or IS-IS is in the range of tens of milliseconds [32] up to several seconds [31], and it provides decreased convergence times to improve a data center's availability at the expense of scalability. Torii directly forgets about convergence times since it has zero unavailability during reconfiguration due to its path diversion capability. Torii's mechanism used for dealing with failures is very similar to the one proposed in [15], which can be considered the state-of-the-art technique to recover from network failures in data centers. For that reason, and in addition to the explanations given in section 3.3, we can conclude that from a fault tolerance perspective, Torii is expected to behave very well.

Regarding the origins of HLMACs, Hierarchical MAC addresses, as a way to circumvent the scalability restrictions of flat Ethernet addresses, they were first proposed in UETS [33]. Hierarchical addresses with tree-based topological significance are used in HURP [10] and other protocols [8]. Another examples are MOOSE [34] protocol and Path-Moose [35], which use locally assigned hierarchical addresses based on a bridgeID:hostID structure. BridgeID in MOOSE protocols must be assigned by a separate protocol that must ensure unique BridgeID assignment, and that BridgeID has no topological meaning. However, Torii is the first protocol that uses multiple tree-based addresses.

7. CONCLUSION

Torii is a simple and efficient layer-two protocol for data center networks. Torii is fully distributed, given that multiple addresses are automatically assigned without duplicates with no need of a centralized address manager module, being this centralized module a requirement in PortLand. Torii also accepts a wider range of topologies than other data center architectures and allows simpler wiring, which enhances scalability and flexibility for network management. Another advantage regarding scalability is that routing and path repair is performed based solely on the destination tree-based HLMAC address used, without requiring routing tables at switches, allowing high speed forwarding. In case of a link failure in a path, the bridge instantly selects an alternative path to reach

Trans. Emerging Tel. Tech. 2014; **00**:1–20
DOI: 10.1002/ett
Prepared using ettauth.cls

Table II. Comparison among different routing protocols for data center networks.

Protocol	Network topologies	Addressing and Message Exchange	Tags, Encapsulation	Forwarding State	Repair	Scalability
PortLand	Multi-rooted tree (k-pod based)	Topological address Location Discovery Protocol • Periodic	MAC Address translation at edge bridges	$O(\#ports)$	Managed by Fabric Manager	Centralized Fabric Manager k-pod based model (complex wiring)
VL2	Clos topologies	Topological address Link-state routing protocol • Periodic	IP Address translation at ToR bridges	$O(\#hosts)$	Managed by VL2 Directory System	Distributed VL2 Directory System Load balancing mechanism
DCell	DCell network	Topological address DCell-Routing and Local link-state routing protocol (servers also forward packets) • Periodic	DCN protocol header	$O(\#hosts)$	Local rerouting and Local link-state routing	Fully distributed Complex wiring
BCube	BCube network	Topological address BCube-Routing and neighbor maintenance protocol • Periodic	Path encoded in packet header	$O(\#ports)$	Neighbor maintenance protocol (several seconds)	Fully distributed Active probing for load balancing
Diamond	Diamond network	Topological address Link information exchange • Periodic	IP Address translation at edge bridges	$O(\#ports)$ + $O(\#\&location\ of\ link\ failures)$	Broadcasting link failure info to all switches	Fully Distributed Complex wiring
F10	Multi-rooted tree (k-pod based)	Topological address Location Discovery Protocol • Periodic	MAC Address translation at edge bridges	$O(\#ports)$	On-the-fly path diversion (zero time)	Centralized Fabric Manager k-pod based model (complex wiring)
Torii	Multi-rooted tree and fat trees with pods (no k-pod based required)	Multiple tree-path coded addresses with Extended RSTP • Exchanged just once	MAC Address translation at edge bridges	$O(\#core\ switches)$	On-the-fly path diversion (zero time)	Fully distributed Broadcast reaching hosts (mitigated with ARP proxy)
TRILL	Arbitrary	Link-state routing protocol (extended IS-IS) • Periodic	Extra encapsulation Per hop retag	$O(\#hosts)$	Link-state protocol	Fully distributed
SEATTLE	Arbitrary	Link-state routing protocol • Periodic	None	$O(\#hosts)$	Link-state protocol	Fully distributed

the destination host and also notifies both edge switches serving origin and destination so that the no longer valid path is not chosen again, for a while, i.e. convergence times after repair could be considered zero, which is an optimal situation for data center networks since they require high availability of resources for their communications. 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. Torii is more flexible since it is applicable to many different data center topologies and not as specific as PortLand, VL2, F10, DCell or BCube, while preserving the simplicity of the address assignment. The independence of Torii from IP addresses allows single-IP subnet addressing and maximizes virtual server mobility in data centers. When comparing with SP, results show a very good behavior of Torii in terms of load distribution and packet delays slightly better than SP.

ACKNOWLEDGEMENTS

This work was supported in part by grant from Comunidad de Madrid through Project MEDIANET-CM (S-2009/TIC-1468).

REFERENCES

1. Mysore RN, Pamboris A, Farrington N, Huang N, Miri P, Radhakrishnan S, Subramanya V, Vahdat A. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric, In *ACM SIGCOMM*, 2009.
2. Rojas E, Ibanez G. Torii-HLMAC: A distributed, fault-tolerant, zero configuration fat tree data center architecture with multiple tree-based addressing and forwarding, In *IEEE GLOBECOM*, 2012.
3. Liu L, Ling Z, Zuo Y. Low-delay Node-disjoint Multi-path Routing using Complementary Trees for Industrial Wireless Sensor Networks. *KSII Transactions on Internet and Information System* 2011; **5**(11).
4. Bruni C, Priscoli FD, Koch G, Pietrabissa A, Pimpinella L. Network decomposition and multi-path routing optimal control. *Transactions on Emerging Telecommunications Technologies* 2013; **24**(2):154–165.
5. Vahdat A, Al-Fares M, Farrington N, Mysore RN, Porter G, Radhakrishnan S. Scale Out Networking in the Data Center. *IEEE Micro* 2010; **30**(4): 29–41.
6. Schroeder MD, Birrell AD, Burrows M, Murray H, Needham RM, Rodeheffer TL, Satterthwaite EH, Thacker CP. Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links. *IEEE Journal On Selected Areas in Communications* 1991; **9**(8):1318–1335.
7. Leiserson CE. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transactions on Computers* 1985; **34**(10):892–901.
8. Ibanez G, Garcia-Martinez A, Carral JA, Arco JM, Azcorra A. Evaluation of Tree-based routing Ethernet. *IEEE Communication Letters* 2009; **13**(6):444–446.
9. Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S. VL2: a scalable and flexible data center network, In *ACM SIGCOMM*, 2009.
10. Ibanez G, Garcia-Martinez A, Carral JA, Gonzalez PA, Azcorra A, Arco JM. HURP/HURBA: Zero-configuration hierarchical Up/Down routing and bridging architecture for Ethernet backbones and campus networks. *Computer Networks* 2010; **54**(1):41–56.
11. Elmeleegy K, Cox A. EtherProxy: Scaling the Ethernet by suppressing broadcast traffic, In *IEEE INFOCOM*, 2009.
12. Myers A, Ng TSE, Zhang H. Rethinking the Service Model: Scaling Ethernet to a Million Nodes. In *Third Workshop on Hot Topics in networks (HotNets-III)*, 2004.
13. 802.1Qbg - Edge Virtual Bridging: http://www.ieee802.org/1/pages/802.1bg.html [March 2014].
14. Clos C. A Study of Non-Blocking Switching Networks. *Bell System Technical Journal* 1953; **32**(2):406–424.
15. Liu V, Halperin D, Krishnamurthy A, Anderson TE. F10: A Fault-Tolerant Engineered Network. In *NSDI*, 2013; 399–412.
16. Sun Y, Chen J, Lu Q, Fang W, Diamond: An Improved Fat-tree Architecture for Large-scale Data

DOI: 10.1002/ett
Prepared using ettauth.cls

Centers. *Journal of Communications* 2014; **9**(1):91–98.
17. Guo C, Lu G, Li D, Wu H, Zhang X, Shi Y, Tian C, Zhang Y, Lu S. BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers, In *ACM SIGCOMM*, 2009.
18. Guo C, Wu H, Tan K, Shi L, Zhang Y, Lu S. DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers, In *ACM SIGCOMM*, 2008.
19. Caro LF, Papadimitriou D, Marzo JL. Ethernet label spaces dependency on network topology. *European Transactions on Telecommunications* 2010; **21**(6):491–503.
20. Dijkstra EW. A note on two problems in connexion with graphs. *Numerische Mathematik* 1959; **1**:269-271.
21. OMNeT++: http://www.omnetpp.org/ [March 2014].
22. INET Framework: http://inet.omnetpp.org/ [March 2014].
23. Prasad RS, Dovrolis C. Beyond the Model of Persistent TCP Flows: Open-Loop vs Closed-Loop Arrivals of Non-persistent Flows, In *41st Annual Simulation Symposium (ANSS)*, 2008; 121–130.
24. Kvalbein A, Dovrolis C, Muthu C. Multipath load-adaptive routing: Putting the emphasis on robustness and simplicity, In *17th IEEE International Conference on Network Protocols (ICNP)*, 2009; 203–212.
25. Perlman R, Eastlake D, Dutt DG, Gai S, Ghanwani A. Rbridges: Base Protocol Specication. *Technical report*, Internet Engineering Task Force, 2009.
26. Kim C, Caesar M, Rexford J. Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises, In *ACM SIGCOMM*, 2008.
27. Al-Fares M, Loukissas A, Vahdat A. A Scalable, Commodity, Data Center Network Architecture, In *ACM SIGCOMM*, 2008.
28. Chen K, Guo C, Wu H, Yuan J, Feng Z, Chen Y, Lu S, Wu W. Generic and Automatic Address Configuration for Data Center Networks, In *ACM SIGCOMM*, 2010.
29. Hu Y, Zhu M, Xia Y, Chen K, Luo Y. GARDEN: Generic Addressing and Routing for Data Center Networks, In *IEEE 5th International Conference on Cloud Computing*, 2012; 107–114.
30. Walraed-Sullivan M, Mysore RN, Tewari M, Zhang Y, Marzullo K, Vahdat A. ALIAS: Scalable, Decentralized Label Assignment for Data Centers, In *ACM SOCC*, 2011.
31. Walraed-Sullivan M, Marzullo K, Vahdat A. Scalability vs. Fault Tolerance in Aspen Trees. *Microsoft Research technical report*, MSR-TR-2013-21, February 2013.
32. Miercom. Lab Testing Summary Report. September 2011 (report 111013). Product Category: Ethernet Fabric. Products Tested: Shortest Path Bridging (SPB) Protocol: http://www.miercom.com/pdf/reports/20111027.pdf [March 2014].
33. Morales J, Ibanez G. Ethernet Fabric Routing (UETS/EFR) - A Hierarchical, Scalable and Secure Ultrahigh Speed Switching Architecture, In *THCSN conference INFOCOM*, 2006.
34. Scott M, Crowcroft J. MOOSE: Addressing the scalability of Ethernet, In *Eurosys Poster section*, 2008.
35. Ibanez G, Marsa-Maestre I, Lopez-Carmona MA, Perez-Ibanez I, Tanaka J, Crowcroft J. Path-Moose: A Scalable All-Path Bridging Protocol. *IEICE Transactions on Communications* 2013; **E96-B**(3):756–763.

AUTHORS' BIOGRAPHIES

Elisa Rojas received her M.S. and Ph.D. in Communication and Information Technologies engineering from the University of Alcala, Spain, in 2011 and 2013 respectively. She worked at Telefonica I+D for 3 years and then joined the Telematics Engineering area of the University of Alcala in 2010, where her research interests are mainly high performance and scalable Ethernet networks, especially focused on data center networks. She is author of several publications on these subjects, some of them about the ARP-Path and Torii-HLMAC protocols.

Guillermo Ibanez received his Telecommunications Engineering degree from Universidad Politécnica de Madrid in 1975 and the Ph.D. in Communication Technologies from Universidad Carlos III de Madrid in 2005. He worked at IT&T R&D Labs and at Alcatel

Telecommunications in switching and access systems. He is currently an associate professor in the Telematics Engineering area if the Universidad de Alcalá in Madrid. He is author of multiple publications and patents on Advanced Ethernet switching.

Jose Manuel Gimenez-Guzman received his M.S. and Ph.D. in Telecommunication Engineering from the Universitat Politècnica de València (UPV), Spain, in 2002 and 2008 respectively. In 2009, he received an Extraordinary Doctorate Award from UPV. Since 2008 he has been at the University of Alcalá, Spain, where he is currently an associate professor. He is engaged in research and teaching in the areas of analysis and performance evaluation of wireless networks. He has published more than 30 technical papers in these areas in international journals and conference proceedings and acts regularly as a reviewer for refereed journals and conferences.

Diego Rivera received a B.S. degree in Computer Science and M.S. in Information Technologies and Communications from the University of Alcalá in 2010 and 2013 respectively. He is currently a Ph.D. candidate and research staff at the GIST (Telematics Engineering) research group at University of Alcalá. His research interests include computer network architectures, algorithms and protocols, intelligent agents and artificial intelligence applied to computer networks.

Arturo Azcorra received his M.Sc. degree in telecommunications engineering from UPM in 1986 and his Ph.D. from the same university in 1989. In 1993 he obtained an M.B.A. from the Instituto de Empresa. He holds a double appointment as full professor (with chair) at the Telematics Engineering Department of University Carlos III of Madrid and director of Institute IMDEA Networks where he conducts his research activities. He has participated in and directed 49 research and technological development projects, including European ESPRIT, RACE, ACTS, and IST programs. He has served as a Program Committee member in many international conferences, including several editions of IEEE PROMS, IDMS, QofIS, CoNEXT, and IEEE INFOCOM. He has published over 100 scientific papers in books, international magazines, and conferences.

DOI: 10.1002/ett
Prepared using ettauth.cls