INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2015; **00**:1–19
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nem

Software Defined Network Based Data Center Architecture for the Reduction of Layer-2 Broadcast Traffic

Veena S*, G. Vijay Teja, Karishma Sureka, Sai Gopal, Ram P. Rustagi, K. N. B. Murthy
P.E.S.University,Bangalore,India

100 ft Ring Road, Banashankari II stage, Bangalore, Karnataka, India

SUMMARY

It is common for user applications to span over thousands of servers, running in several virtual machines spread across different Data Centers. A single user search request might access these servers several times for the data. In addition, Data Centers are now becoming multi-tenant which increases the amount of communication within the Data Center. Most Data Centers are based on a Multi-rooted, 3-tier topology with inbuilt redundancy. In such topologies, internal traffic management is very much necessary. This can be achieved by designing the network as a single L2 network rather than the traditional L3 network with the former having several advantages like ease of administration, increased throughput, reduced latency etc. However, large L2 networks have severe scalability issues due to the large amount of broadcast traffic that they generate. This paper addresses the requirements of a scalable and efficient L2 network, especially for Data Centers where the internal traffic performance is critical, by the use of an SDN (Software Defined Network) application. The global network topology information available at the SDN Controller, is used to assign hierarchical, positional pseudo MAC addresses to serve as host locators within the network. This enables the controller to insert high level forwarding policies that make use of the structured addressing scheme in all the network switches and reduce the L2 switch broadcast traffic. The SDN controller is utilized to intercept ARP broadcast requests and eliminate them whenever the requested information is centrally available. Our results show that there is a significant performance boost in the internal communication for L2 based Data Center networks after the introduction of our SDN application.

Received . . .

KEY WORDS: ARP broadcast; Data Centers; MAC Address; Openflow; SDN Controller; Virtual Machines

1. INTRODUCTION

With the advent of Server virtualization, a single physical machine can support multiple Virtual Machines (VMs). Each of these VMs can host several cloud services. With this architecture, we have 10s to 100s of services offered by a single physical server. Millions of users interact with these

*Correspondence to: Dept. of MCA, 100 ft Ring Road, Banashankari II stage, Bangalore, Karnataka, India, E-mail: sveena@pes.edu

servers to fetch the required data [3]. In Data Centers all these servers are installed in a rack and connected to the TOR (Top of the Rack) switch through their Ethernet interfaces. There may be 100s of such racks and these TOR switches are connected to the EOR (End of Row) switch which is also called the aggregate switch. These aggregate switches are connected to one or more core switches[5],[9]. This Multi-rooted 3-tier architecture of Data Center creates a very large network within the Data Center and whenever user requests for some service it creates lots of traffic within the Data Center.

Legacy Data Centers used to have more of north-south traffic, i.e., the traffic that moves between the client and the servers, rather than the east-west traffic, i.e., the traffic that moves between the servers and the VMs. Recently the growth of virtualization and cloud computing technologies have resulted in increased east-west traffic in Data Centers [14],[11]. According to the current technical reports there is 76% of east-west traffic, 17% of north-south traffic and 7% of inter Data Center traffic [13]. The inter Data Center traffic is used for disaster recovery purposes and it is growing gradually.

As the traffic within the Data Center is growing, proper administration and maintenance is necessary. Administration becomes easy if it is viewed as a L2 network rather than a L3 network [1], as L3 networks have lot of configuration requirements. But L2 networks have scalability issues and generate lot of broadcast traffic. In our proposed work, efforts are made to reduce this broadcast traffic by the introduction of hierarchical pseudo MAC addresses which are based on the position of the host in the network. Our work also improves the performance of the Data Center network by utilizing the available redundancy in the topology of the network and eliminating the usage of the spanning tree protocol while communicating at Layer-2.

In a normal L2 network, whenever a host wants to communicate to another host whose MAC address is not known, an ARP request is generated, which is likely to result in multiple broadcast traffic. In the proposed work the concept of SDN is used to eliminate this type of traffic. In SDN, the control plane is decoupled from the data forwarding plane [15] and the full topology of the network is known to the central controller. Whenever a host request for the MAC address of an unknown host, the request goes to the SDN controller and it generates the ARP reply as if it is coming from the destination host. By this way we can reduce the ARP Broadcast traffic.

To evaluate the performance, the traffic statistics in the Data Center network were compared in the presence and absence of our SDN application. The results indicate that there is a drastic reduction in broadcast traffic and a consequent high performance gain with the introduction of our application in SDN based Data Centers.

1.1. Problem Description

Administration and maintenance of Data Centers becomes simple if it is viewed as an L2 Network rather than L3 network for the following reasons. In L3 networks every element of the network need to be configured with the IP address indicating the subnet to which it belongs to, in addition to the address of the gateway router. Sometimes DHCP servers need to be synchronized to distribute the IP addresses. Each packet in a network element (e.g. router) need to be processed for an extra layer-3 header. As low latency is of high priority in Data Centers, it is better to maintain it as an L2 network than an L3 network. But L2 networks have lot of scalability issues in large networks.

DOI: 10.1002/nem

L2 switches are of self learning in nature. i.e., an L2 switch learns the location of the MAC address through the packets that pass through it. It learns about the port and MAC address and stores this information in a table called the MAC table. Whenever a switch receives a new packet it forwards the packet as per the information available in the MAC table. If the network is very active then this table grows very fast which slows down the MAC lookup process and increases the latency. Lookup latency could be decreased by the use of CAM (Content Addressable Memory), but it results in increased cost[2].

There are several issues with L2 networks in Data Centers.

1.1.1. Unknown unicast: When a new packet arrives at the switch, if the MAC table does not have an entry for the destination MAC, then the packet is flooded through all the ports except the source port. This results in broadcast of traffic which is exponentially proportional to the size of the network.

1.1.2. Domain Broadcast: Some higher level protocols like DHCP (Dynamic Host Configuration Protocol), ARP (Address Resolution Protocol) etc. rely on broadcast mechanism for their operations. Here the packet need to be delivered to everybody in the network.

1.1.3. Necessity for Creation of Spanning Tree: Data Center networks usually have redundant links to handle link failures. These redundant links create loops in the network and it is necessary to run the spanning tree protocol to avoid broadcast storms. But the spanning tree creates a single tree for the whole large network. While communicating between two nodes in the network, if we follow the path guided by the spanning tree, it may be a suboptimal, longer path. Also, it may clog the network near the root of the spanning tree.

The above mentioned issues create a lot of broadcast traffic in large networks like Data Centers. It also consumes lot of network bandwidth and increases the latency.

1.2. Proposed Methodology

The aim of our work is to address the above mentioned issues with respect to large L2 based Data Center networks. Even though L2 networks meet many of the changing requirements of Data Centers like ease of administration and VM Migration (with easier IP mobility within an L2 network), scalability issues prevent Data Centers from deploying large scale L2 Networks. The unknown unicast happens because of the flat addressing nature of MAC addresses. Since the destination host could be anywhere in the network, it is helpful to introduce some sort of structure to the MAC address [1] to handle this issue. We are introducing this structure to the MAC in the form of Pseudo-MAC to identify the host in the network based on its location in the network. This hierarchical Pseudo MAC structure helps in reducing the broadcast traffic due to unknown unicast in L2 Networks.

The ARP Broadcast traffic can be handled with the introduction of the SDN concept which is gaining a lot of importance in the world of virtualization [7]. There will be a central controller which controls the forwarding elements called switches by dictating the flow entries into the flow tables, an equivalent of MAC tables in regular L2 networks. Communication between the controller and the switches happens through the openflow protocol [12]. It allows the network to be programmed on

DOI: 10.1002/nem

a per-flow basis and provides extremely granular control over the network without any specialized hardware. This level of control is not possible in current IP-based routing. Due to this paradigm shift, SDN has witnessed a wide audience from the enterprise space and hence, it is desirable to solve the scaling issue in the context of an SDN based architecture in Data Center networks.

In L2 networks, communication between two hosts depends on the spanning tree protocol to select the best path whenever multiple paths are available. In the proposed work, the knowledge of the topology information available at the controller is intelligently used to eliminate the requirement of the spanning tree protocol. The redundancy available in the Data Center network is further leveraged to efficiently load balance the traffic between the multiple paths instead of always selecting the same path between the two endpoints. This increases the bandwidth utilization within the Data Center network and consequently improves the throughput.

The proposed work shows a performance improvement for a 3-tier multi-rooted Data Center network architecture which uses an out of band SDN controller. It creates a separate control network between the network switches and the SDN controller. The controller builds the global topology information received through link-up and switch-up events on network setup. This information is used to identify the different levels of switches namely - core, aggregate and edge switches. This information also helps in dividing the address space for the assignment of hierarchical pseudo MAC addresses. The controller inserts static forwarding entries into the flow tables of all the switches to forward the data packets based on the pseudo MAC addresses. To ensure that the pseudo addressing scheme used internally is transparent to the hosts, translation entries are inserted into the corresponding edge switches for every new host in the network. Unknown unicast is completely eliminated as the pseudo MAC addresses serve as host locators within the Data Center network, and the controller intercepts ARP queries to reduce the domain broadcast traffic. The forwarding entries inserted in the switches also utilize the inbuilt redundancy in the Data Center network links and eliminate the necessity of the spanning tree protocol.

2. SYSTEM DESIGN

The goal of our work is to reduce the L2 broadcast traffic in SDN based Data Center network. Efforts are made to reduce the switch level broadcast and the ARP broadcast and also load balance the traffic by using the built in redundancy available in the Data Center network. The centralized SDN controller which can communicate to all the switches in the network to give the flow table information is considered for the proposed work. Distributed nature of communication which involves controller to controller communication is out of the scope for our work. We are leveraging on the observation that the Data Centers use the multi-rooted 3-tier topology which is relatively fixed and can be upgraded by the introduction of SDN controller to improve the performance.

2.1. Architecture of SDN based Data Center

For the implementation of the SDN based Data Center network, a multi-rooted 3-tier topology as shown in Figure1 is considered. Many Data Centers uses this topology for their implementation. The architecture of the proposed system is as shown in Figure2.

DOI: 10.1002/nem

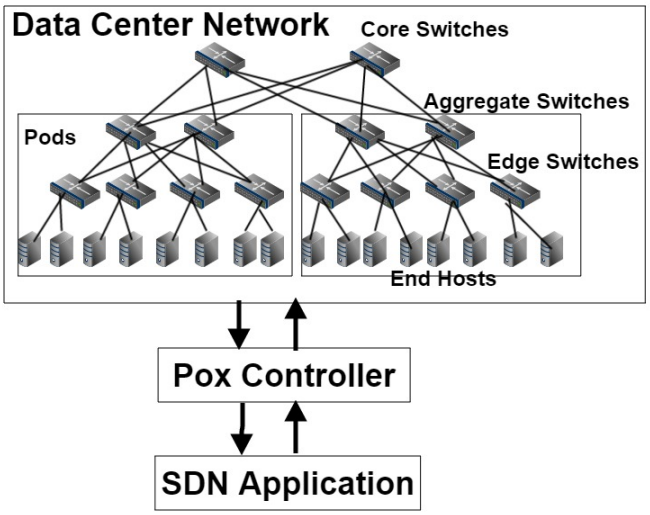


Figure 1. Multi-rooted 3-tier Topology of Data Centers.

2.1.1. Openflow Enabled Network: A number of end hosts on which the VMs running the user applications resides are connected to the edge switches, and these edge switches are connected to the aggregate switches and they in turn connect to the core switches. All these switches are openflow enabled and communicate with the SDN controller using the openflow protocol. One can observe the redundancy available between the core switches to the aggregate switches and aggregate switches to the edge switches. If the links between the core and aggregate switches are removed from the topology, it results in sub-graphs (excluding the core switches), each comprises of a set of aggregate switches, edge switches and their connected hosts. Each such sub-graph is identified as a pod, as shown in Figure1 and the pseudo MAC address will be given to every host based on the position of the host within the pod. Each end host may accommodate several virtual machines(VMs) and can be identified with their unique IDs. The translation of pseudo MAC to actual MAC and vice versa is taken care by the edge switches which results in the abstraction of actual MAC completely within the Data Center network.The proposed application runs on top of a standard SDN controller and takes care of normal functioning of networks. The unicast data packets will be forwarded as in normal networks but the broadcast traffic will be reduced to increase the performance.

2.1.2. SDN Controller: The central part of the SDN Based Data Center is the SDN Controller. It is the entity that directly interacts with the switches (via the Southbound interface such as openflow) and controls the networks operations by dictating how, where and when the switches forward the data packets they receive. The controller also acts as a platform that provides network applications with a base set of network functions (such as Discovery) and provides APIs (Northbound API) that the applications can use to make use of the network functions, gain access to the control information available at the controller and also interact with the underlying network. Once initial handshaking is done the controller sends the feature request message to all switches and extracts their DPIDs (Data Path IDs). With this it can uniquely identify all switches and control their operations.

Prepared using nemauth.cls
DOI: 10.1002/nem

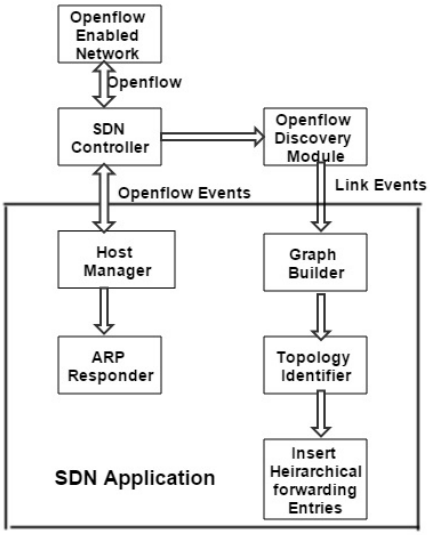


Figure 2. SDN Based Data Center Architecture

2.1.3. Openflow Discovery Module: Network discovery is one of the base network functions that is provided by the controller. Controller does this through the discovery module and is responsible for monitoring the underlying network and detecting link changes (addition or removal of a link). Any application can subscribe to this module and be notified about link changes. The discovery module instructs switches to flood out specialized LLDP (Link Layer Discovery Protocol) packets, with information about the source switch - its DPID and port, through all the links going out from the switch. Once a switch on the other end of the link receives these packets, it generates a PacketIn event, and the event generated has the switch DPID and port information of either end of the link. The discovery module makes use of this information to maintain link information and generate link events on changes.

2.2. SDN Application

2.2.1. Graph Builder: This module is responsible for building the network map of the whole Data Center network by listening to the Link events and Connection-Up events generated by the Discovery module and the Controller respectively. This graph is further consulted by several of the sub-modules to perform their functions.

2.2.2. Topology Identifier: This sub-module has the following two roles:

Switch classification: Using the network view built by the Graph builder, all the switches need to be classified as edge, aggregate and core switches. Here we are leveraging on the assumption that the number of end hosts connected to an edge switch must be more than the number of aggregate switches it is connected to. First, all the edge switches are identified based on this assumption. Then, all the switches directly connected to the edge switches are identified as aggregate switches

DOI: 10.1002/nem

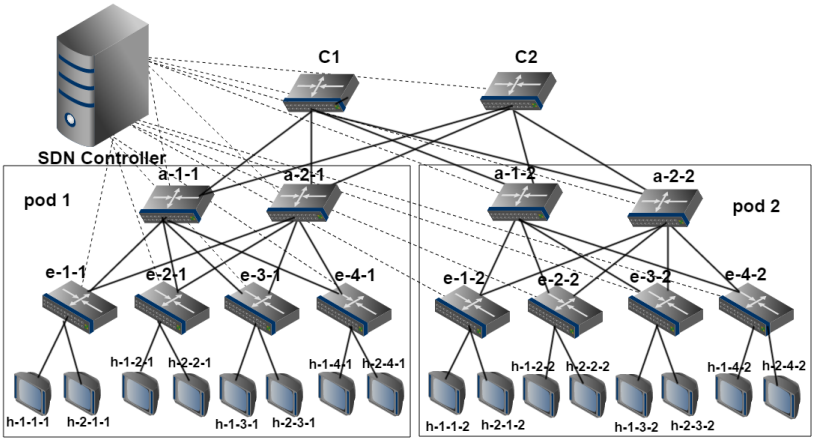


Figure 3. Labelled Topology Diagram

and the remaining switches as core switches.

Pod identification: Now these switches need to be grouped into different pods to facilitate the hierarchical pseudo MAC address structure. In the network view, the links from aggregate to core switches are removed logically and the resulting sub-graphs are identified as pods. By traversing these sub-graphs using standard graph traversal algorithms like Breadth First Search (BFS), one can identify the pods uniquely and also fix the positions of the switches within the pod uniquely.

The switches and the pods are given the unique numbers as shown in Figure3. The core switches are given the labels based on their position starting from 1, for aggregate and the edge switches the first number from right indicate the pod number and the next indicate the position within the pod. For the end host, the first number from right indicate the pod number and the second number indicate the edge switch to which this host is connected. The third number indicate the position of the host connected to that edge switch. This pod number and position information are used while forming the pseudo MAC addresses.

2.2.3. Inserting Forwarding Entries: All the forwarding actions in the proposed architecture relies on the following hierarchical addressing scheme of the pseudo MAC address. The format of 48 bit Pseudo MAC address is as follows - The most significant 16 bits are used to indicate the pod number in which the host is located, the next 8 bits hold the switch number (within the pod) of the edge switch that the host is connected to followed by 8 bits of port number within that edge switch and the least significant 16 bits are used to hold the VMID. There could be multiple VMs (each having their own IP and MAC) running on a single physical host and all of them would be connected to the network using the same edge switch and port. To differentiate between multiple such VMs and assign unique pseudo MACs to each of them, a 16-bit VMID that is unique within an edge switch and port pair was introduced in the proposed architecture. For example, the pseudo MAC address of host h1-1-1 is 00:01:01:03:00:01 (pod-1, edge switch-1,port-3 and VMID-1) and that of h2-1-1 is 00:01:02:04:00:01 (pod-1, edge switch-1, port-4 and VMID-1) and so on.

Prepared using nemauth.cls
DOI: 10.1002/nem

Having complete knowledge of both the topology of the network and the encoding of host position into the pseudo MAC address, the application is capable of inserting the flow table entries to all switches proactively for normal data forwarding. If there are multiple ways to reach a destination, simple bit mask hashing is done to utilize redundant links and load balance the flows between them. Unknown unicast and broadcast packets would be handled differently by different category of switches. Unicast forwarding could be handled as follows.

Core Switch: The core switches connect to all the pods and hence their forwarding goal is to just forward the incoming packet to the correct pod. It achieves this by looking at the pod number (first 2 bytes) of the destination PMAC to find the corresponding pod and forwards the packet to one of the aggregate switches in that pod.

Aggregate Switch: First the pod number of the destination PMAC is examined to find out if the received packet should be forwarded within/outside the pod. If it is in the same pod then the edge switch position (3rd byte) of the destination PMAC is examined and the packet is forwarded directly to that edge switch. Otherwise the packet is forwarded to one of the core switches above for further forwarding.

Edge Switch: First, the most significant 24 bits of the destination PMAC are examined to find out if both the pod and edge switch number match with its own. If so then the next 8 bits of the destination PMAC are examined to find the port number to identify the right destination host, and also it translates the pseudo MAC to actual MAC before forwarding the packet to the respective host. Otherwise the packet is forwarded to one of the aggregate switches above for further forwarding.

Thus Unknown unicast is completely avoided as there are entries to match every entry in the MAC address space. A sample flow table entry is as shown in Figure4. Each flow table entry follows the structure as shown in Figure5. There would be several match fields (Header fields) against which the incoming packet is matched. Bit masks were used to examine specific portions of the pseudo MAC address and match against them. If there is a match, the corresponding actions in the flow table entry will be executed. If multiple matches are there then higher priority entry gets the preference.

Normal L2 networks uses the spanning tree protocol to avoid broadcast storms as there are a lot of loops in the topology. In the proposed work the ports are classified into up ports and down ports and insert the flow table entries into the switches to perform controlled broadcast.

Controlled Broadcast: In core switches there are only down ports, and it should not forward to the port connected to the same source pod from where it receives the packet. Also for the remaining pods it should forward only once for each pod. This is achieved using the flow hash based on the destination MAC port number available in the fourth byte of the pseudo MAC address.

In case of aggregate switches and edge switches while sending the traffic upwards it uses the flow hash based on the destination MAC port number available in the fourth byte of the pseudo MAC address as it should forward the packet only to one of the up links. While sending the traffic downwards it floods the packet through all its down ports (avoiding down source port if any). Thus broadcast storms are effectively avoided without the use of spanning tree protocol.

DOI: 10.1002/nem

```
cookie=0x0, duration=2671.201s, table=0, n_packets=0, n_bytes=0,
idle_age=2671,
priority=2000,dl_dst=00:00:00:00:00:00/00:00:00:01:00:00
actions=output:1

cookie=0x0, duration=14.454s, table=1, n_packets=2, n_bytes=196,
idle_age=14,
priority=5000,dl_dst=00:00:01:03:00:01
actions=mod_dl_dst:46:d9:a2:6d:d5:94,
output:3

cookie=0x0, duration=2671.304s, table=2, n_packets=0, n_bytes=0,
idle_age=2671,
priority=3000,dl_dst=00:00:01:04:00:00/ff:ff:ff:ff:00:00 actions=output:4
```

Figure 4. Sample Flow-Table Entries

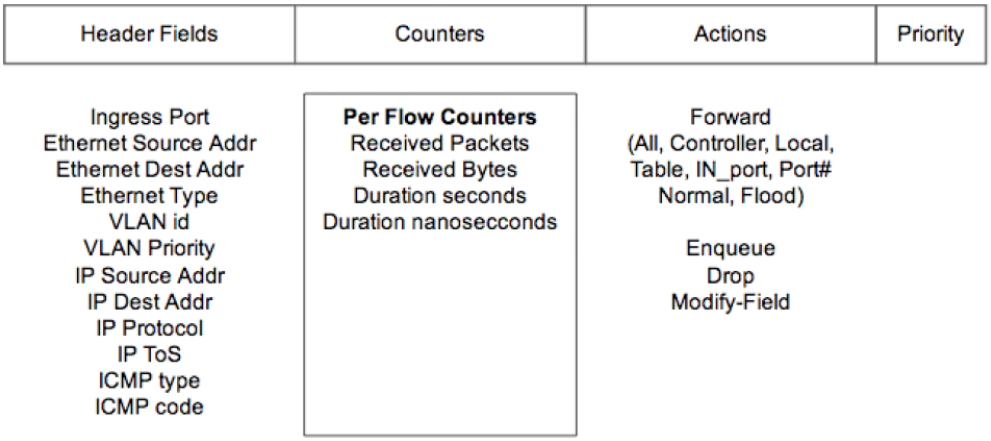


Figure 5. Flow-Table

For example, if it is required to verify whether the destination PMAC is within pod 2 or pod 4, two flow table entries would be created, both having a mask of FF:FF:00:00:00:00 on the destination address. This would ensure that only the pod number gets examined. Next it is required to have one entry to match for pod number 2 and another for pod number 4. It is achieved by matching with 00:02:00:00:00:00 in case of first entry and with 00:04:00:00:00:00 for the second entry. The order of matching can be achieved by assigning different priority values. The highest priority entry would get the preference of execution.

2.2.4. Host Manager: Every new host in the network must be discovered and assigned a positional pseudo MAC before it could do any communication within the network. The host manager is responsible for performing both the aforementioned responsibilities.

Host discovery - The host manager transparently and passively discovers new hosts in the network by relying on the assumption that whenever a host become active and reachable, it sends a gratuitous ARP message advertising its IP and MAC address. Special flow table entries are inserted in all the edge switches to intercept all ARP messages originating from the hosts, and redirect them to the

DOI: 10.1002/nem

controller in the form of PacketIn message. The Host Manager in our application discovers the host and the port of the switch that the host connects to, and helps in the creation of the pseudo MAC address.

Pseudo MAC assignment - Whenever a new host is discovered, a new pseudo MAC is generated based on the hierarchical addressing scheme described above, and gives this information in the form of PacketOut message to the respective switches. The Host manager ensures that this pseudo MAC addressing scheme is completely transparent to the hosts with the help of special flow table entries called 'translation entries', that are inserted into the edge switches immediately after the creation. The entries in the edge switches are split into 3 tables that form a pipeline starting from table 0 to facilitate translation. Table 0 is responsible for intercepting ARP packets coming from hosts and converting the actual MAC to pseudo MAC while sending the data. Table 1 is responsible for translating the pseudo MAC to actual MAC while receiving the data and forwarding the packet to the right port.Table 3 is responsible for load balancing and forwarding the packet to one of the aggregate switches above. The 2nd and 3rd entries in Figure4 are examples of translation entries.

2.2.5. ARP Responder: In normal networks, the ARP requests are always broadcasted and creates lot of broadcast traffic. The proposed work reduces this by the ARP responder module. Any ARP packet comes from the end host is redirected to the controller. If the packet is an ARP request, it verifies the IP - PMAC table and performs a look-up for the destination IP the ARP request is querying for. If it is available, it fabricates an ARP response with the PMAC of the destination IP that was queried and send it to the edge switch of the requesting host. Otherwise a new ARP request message is fabricated by replacing the source MAC address with the corresponding PMAC address, and broadcasting this to all the edge switches and in turn to all the hosts in the network. If the packet is an ARP response, the module replaces the source MAC address with the corresponding PMAC and then sends the response as a PacketOut directly to the edge switch of the destination host.

The different modules in the proposed architecture completely eliminates the unknown unicast traffic, reduces the domain broadcast traffic and also eliminate the broadcast storms without using the spanning tree protocol.

2.3. A Complete End-to-End Example

Consider an example of host h-1-1-1 (original MAC - 46:d9:a2:6d:d5:94) connected to e-1-1 wants to ping the host h-1-1-2 (original MAC 3a:57:3d:b2:78:be) connected to e-1-2 (refer to the topology diagram in the Figure3).

On the start of the application, the topology of the network is discovered and forwarding entries are inserted into the respective switches as in Figures6 and 7. As and when the hosts become reachable in the network, they send gratuitous ARP announcing their arrival and the flow table entries 3 and 4 in the flow tables of e-1-1 and e-1-2 instruct the switch to redirect these ARP packets coming from host ports, directly to the controller. The Host manager in our application immediately assigns unique pseudo MAC addresses to them and adds the corresponding translation entries to the edge switches. Entries 2 and 6 in flow tables of e-1-1 and e-1-2 are the translation entries of hosts h-1-1-1 and h-1-1-2 that are inserted after discovery.

Initially, h-1-1-1 would send an ARP query requesting for h-1-1-2's MAC address. This query is intercepted at the edge switch e-1-1 and sent to the controller by the flow table entries 3 and 4 in

DOI: 10.1002/nem

Edge Switch e-1-1

1 cookie=0x0, duration=2671.279s, table=0, n_packets=2, n_bytes=196, idle_age=14, priority=10, actions=resubmit(,1)

2 cookie=0x0, duration=14.455s, table=0, n_packets=2, n_bytes=196, idle_age=14, priority=5000, in_port=3, dl_src=46:d9:a2:6d:d5:94, actions=mod_dl_src :00:00:01:03:00:01, actions=resubmit(,1)

3 cookie=0x0, duration=2671.279s, table=0, n_packets=1, n_bytes=42, idle_age=14, priority=9000, arp, in_port=3, actions=CONTROLLER:65535

4 cookie=0x0, duration=2671.279s, table=0, n_packets=0, n_bytes=0, idle_age=2671,

priority=9000, arp, in_port=4, actions=CONTROLLER:65535

5 cookie=0x0, duration=2671.279s, table=1, n_packets=2, n_bytes=196, idle_age=14, priority=10, actions=resubmit(,2)

6 cookie=0x0, duration=14.454s, table=1, n_packets=2, n_bytes=196, idle_age=14, priority=5000,dl_dst=00:00:01:03:00:01 actions=mod_dl_dst:46:d9:a2:6d:d5:94, output:3

7 cookie=0x0, duration=24.978s, table=2, n_packets=0, n_bytes=0, idle_age=24, priority=2000, dl_dst=00:00:00:00:00:00/00:00:00:01:00:00, actions=output:1

8 cookie=0x0, duration=24.978s, table=2, n_packets=2, n_bytes=196, idle_age=9, priority=2000, dl_dst=00:00:00:01:00:00/00:00:00:01:00:00 actions=output:2

(a) Sample Flow Table entries of e-1-1

Edge Switch e-1-2

1 cookie=0x0, duration=24.956s, table=0, n_packets=2, n_bytes=196, idle_age=9, priority=10, actions=resubmit(,1)

2 cookie=0x0, duration=10.068s, table=0, n_packets=2, n_bytes=196, idle_age=9, priority=5000, in_port=3, dl_src= 3a:57:3d:b2:78:be, actions=mod_dl_src :00:01:01:03:00:01, resubmit(,1)

3 cookie=0x0, duration=24.956s, table=0, n_packets=2, n_bytes=84, idle_age=4, priority=9000, arp, in_port=3, actions=CONTROLLER:65535

4 cookie=0x0, duration=24.956s, table=0, n_packets=0, n_bytes=0, idle_age=24, priority=9000, arp, in_port=4, actions=CONTROLLER:65535

5 cookie=0x0, duration=24.948s, table=1, n_packets=2, n_bytes=196, idle_age=9, priority=10, actions=resubmit(,2)

6 cookie=0x0, duration=10.033s, table=1, n_packets=2, n_bytes=196, idle_age=9, priority=5000, dl_dst=00:01:01:03:00:01,actions=mod_dl_dst:3a:57:3d:b2:78:be, output:3

7 cookie=0x0, duration=24.978s, table=2, n_packets=0, n_bytes=0, idle_age=24, priority=2000, dl_dst=00:00:00:00:00:00/00:00:00:01:00:00, actions=output:1

8 cookie=0x0, duration=24.978s, table=2, n_packets=2, n_bytes=196, idle_age=9, priority=2000, dl_dst=00:00:00:01:00:00/00:00:00:01:00:00, actions=output:2

(b) Sample Flow Table entries of e-1-2

Figure 6. Edge-Switch Flow Tables

the flow table of e-1-1. The interception happens prior to the translation of original MAC to pseudo MAC because ARP interception is at a higher priority. One can observe in the Figure6, both the flow table entries 2 and 3 will match, but the entry 3 gets executed first as it is having higher priority compared to the entry 2. At the controller, The ARP Responder module receives the ARP query and fabricates an ARP response with h1-1-2's PMAC which is already known to it and send it directly back to h-1-1-1.

Prepared using nemauth.cls
DOI: 10.1002/nem

Aggregate Switch a-2-1

```
1 cookie=0x0, duration=2671.201s, table=0, n_packets=0, n_bytes=0, idle_age=2671,
priority=2000, dl_dst=00:00:00:00:00:00/00:00:00:01:00:00, actions=output:1

2 cookie=0x0, duration=2671.201s, table=0, n_packets=2, n_bytes=196, idle_age=14,
priority=2000, dl_dst=00:00:00:01:00:00/00:00:00:01:00:00,  actions=output:2

3 cookie=0x0, duration=2671.201s, table=0, n_packets=0, n_bytes=0, idle_age=2671,
priority=3000, dl_dst=00:00:03:00:00:00/ff:ff:ff:00:00:00,  actions=output:5

4 cookie=0x0, duration=2671.201s, table=0, n_packets=2, n_bytes=196, idle_age=14,
priority=3000, dl_dst=00:00:01:00:00:00/ff:ff:ff:00:00:00, actions=output:3

5 cookie=0x0, duration=2671.201s, table=0, n_packets=0, n_bytes=0, idle_age=2671,
priority=3000, dl_dst=00:00:02:00:00:00/ff:ff:ff:00:00:00, actions=output:4

6 cookie=0x0, duration=2671.201s, table=0, n_packets=0, n_bytes=0, idle_age=2671,
priority=3000, dl_dst=00:00:04:00:00:00/ff:ff:ff:00:00:00, actions=output:6
```

(a) Sample Flow Table entries of a-2-1

Aggregate Switch a-2-2

```
1 cookie=0x0, duration=4567.827s, table=0, n_packets=0, n_bytes=0, idle_age=4567,
priority=2000, dl_dst=00:00:00:00:00:00/00:00:00:01:00:00, actions=output:1

2 cookie=0x0, duration=4567.827s, table=0, n_packets=2, n_bytes=196, idle_age=4513,
priority=2000, dl_dst=00:00:00:01:00:00/00:00:00:01:00:00, actions=output:2

3 cookie=0x0, duration=4567.827s, table=0, n_packets=0, n_bytes=0, idle_age=4567,
priority=3000, dl_dst=00:01:03:00:00:00/ff:ff:ff:00:00:00, actions=output:5

4 cookie=0x0, duration=4567.827s, table=0, n_packets=2, n_bytes=196, idle_age=4513,
priority=3000, dl_dst=00:01:01:00:00:00/ff:ff:ff:00:00:00, actions=output:3

5 cookie=0x0, duration=4567.827s, table=0, n_packets=0, n_bytes=0, idle_age=4567,
priority=3000, dl_dst=00:01:02:00:00:00/ff:ff:ff:00:00:00, actions=output:4

6 cookie=0x0, duration=4567.827s, table=0, n_packets=0, n_bytes=0, idle_age=4567,
priority=3000, dl_dst=00:01:04:00:00:00/ff:ff:ff:00:00:00, actions=output:6
```

(b) Sample Flow Table entries of a-2-2

Core Switch c2

```
1 cookie=0x0, duration=161.156s, table=0, n_packets=0, n_bytes=0, idle_age=161,
priority=2000, dl_dst=00:00:00:00:00:00/ff:ff:00:01:00:00, actions=output:1

2 cookie=0x0, duration=161.156s, table=0, n_packets=0, n_bytes=0, idle_age=161,
priority=2000, dl_dst=00:01:00:00:00:00/ff:ff:00:01:00:00, actions=output:3

3 cookie=0x0, duration=161.151s, table=0, n_packets=2, n_bytes=196, idle_age=106,
priority=2000, dl_dst=00:01:00:01:00:00/ff:ff:00:01:00:00, actions=output:4

4 cookie=0x0, duration=161.156s, table=0, n_packets=2, n_bytes=196, idle_age=106,
priority=2000, dl_dst=00:00:00:01:00:00/ff:ff:00:01:00:00, actions=output:2
```

(c) Sample Flow Table entries of c2

Figure 7. Flow Table entries for Aggregate and Core Switches

Now h-1-1-1 would like to send an ICMP packet to h-1-1-2, with destination address as the PMAC address of h-1-1-2 obtained in the previous stage. The packet is received by the edge switch e-1-1 and refers to its flow table for further action to be taken. First Table 0 translates the original MAC of h-1-1-1 in the source address of packet to its PMAC as in the flow table entry 2 and resubmit to table 1. The packets passes through table 1 without any change by the flow table entry 5. By using

Prepared using nemauth.cls
DOI: 10.1002/nem

the flow table entry 8 of table 2, the packet is forwarded to the aggregate switch a-2-1. In traditional L2 forwarding, this would have led to an unknown unicast situation. Since e-1-1 would not have h-1-1-2's MAC address in its MAC table, it would have flooded the packet to every other switch connected to it and they in turn floods the packet. Thus a lot of broadcast traffic would be generated.

At a-2-1, the higher priority entries 3-6 checking the packet's destination for pod 0 will fail to match since the destination pod number is not 0. Among the entries 1 and 2, the packet would match only entry 2 and forwarded to c2 by the flow table entry 2.

At c2, packet matches entry 3 and is forwarded to a-2-2. At a-2-2, the packet matches entry 4 and is forwarded to e-1-2 through port 3.

At e-1-2, the packet passes through table 0 without any change since it matches only flow table entry 1. In table 1, the PMAC of h-1-1-2 in the destination address is translated to its original MAC as in entry 6 and forwarded to h-1-1-2 i.e., the final destination through port 3.

Thus, with the help of hierarchical addressing scheme, any packet can be forwarded to the correct location with zero duplication of packet.

3. EXPERIMENTS AND RESULTS

The openflow enabled network was emulated using Mininet. This emulator can be used to create any custom topology of users choice [6], by modifying the Custom.topo file in Mininet. Simplified Mininet is used to create the multi rooted 3-tier topology of Data Center network. To collect the topology information python based modular SDN Controller POX is used. Periodically it provokes all openflow switches to send LLDP and BDDP(Broadcast Domain Discovery Protocol) packets to learn about the links between switches and their broadcast domains.

The scalability issues of L2 networks become apparent with increasing numbers of hosts in the network. Hence, the evaluation methodology was to compare how our architecture fares in comparison to traditional L2 networks with respect to the growing number of hosts.

Comparative evaluation is done based on the following requirements:

- It is necessary to generate a large amount of network traffic and track user performance metrics such as RTT(Round Trip Time). To achieve this, a mininet module - genTraffic, was developed that can generate huge amounts of ping traffic in a controlled manner. It does this by picking two hosts randomly and issuing a ping packet from one host to the other. How often it picks two hosts and for how long it generates traffic can be controlled through parameters passed to the module. For example, *genTraffic 2 -15000 1* means ping requests would be generated in an interval of 2ms, a total of 15000 ping packets would be generated and the number of packets per ping request is 1.
- It is also required to measure the total traffic at the network level that was generated as a result of application traffic, to study the effect on broadcast traffic. For example, one ping request could involve an ARP broadcast and an unknown unicast at most of the switches in the path to destination host. To measure this network traffic, a POX module gstats, was developed. Each Openflow switch maintains different counters. One such counter is the port counter, that measure all the packets that it has transmitted, received, dropped and erroneous. Openflow provides the ability to query these counters on the switches. The gstats module makes use

DOI: 10.1002/nem

Table I. Varying the Number of Hosts

No. of Hosts	No. of Packets Received		No. of Packets Transmitted		RTT in ms	
	With SDN App	Without SDN App	With SDN App	Without SDN App	With SDN App	Without SDN App
32	136494	174244	137115	190384	1.799	43.312
160	150793	499283	167013	2313249	31.125	598.512
320	158081	624804	218606	5531773	63.828	1112.62
480	160041	690533	283890	8409672	92.659	997.729
640	159399	1072277	372801	18943323	99.94	2606.14
800	159342	1019946	497946	20645283	160.32	2102.49

of this ability to query all the switches in the network, to get the port counters of each port of that switch and aggregates these results to get the total amount of network traffic. That is the number of packets received by the switches and the number of packets transmitted by the switches in the network.

Performance of the SDN application can be evaluated by running the POX controller with the proposed application against traditional L2 forwarding. To emulate L2 forwarding, POX provides a built-in module called 'forwarding.l2_learning' which makes the OpenFlow switches behave as layer 2 learning switches. Since the L2 topology has redundant links and loops, spanning tree protocol has to be run along with the l2_learning component to prevent broadcast storms. POX provides a module 'openflow.spanning_tree' to create a centralized spanning tree. Both our SDN app and the spanning tree component require the openflow.discovery module to build a map of the network.

With the help of genTraffic and gstats modules, experiments were conducted to do the comparative study of our application against traditional L2 switch forwarding. Experiments were repeated by varying different parameters as below, to study the behavior of the network with and without our application.

a. Keeping the switch topology constant with each run, vary the number of hosts within the network and collect the results. In each run, using genTraffic, 15000 application(ping) packets were generated and the corresponding RTT was recorded. After generating the traffic, the network activity was measured using the gstats module. The results of these are presented in Table I.

b. Vary the size of the network by varying the number of pods and the number of core switches. The increase in size of the network is in the ratio 1:2:4 for core, aggregate and edge switches respectively. As the number of pods are increasing the number of hosts would also increase. As in the previous case using genTraffic and gstats modules 15000 application packets were generated and the results were gathered with respect to RTT and network traffic. The results are given in Table II.

The evaluation was done on machines with the following configuration: Mininet was emulated on a machine with 4 cores and 64 GB RAM. POX hosting our application was run a machine with 4 cores and 32 GB RAM.

DOI: 10.1002/nem

Table II. Varying the Number Pods and Core Switches

No. of Pods and Core Switches	No. of Packets Received		No. of Packets Transmitted		RTT in ms	
	With SDN App	Without SDN App	With SDN App	Without SDN App	With SDN App	Without SDN App
2, 2	118696	131323	118933	133228	0.519	3.935
4, 4	136449	178594	137248	194396	1.686	47.455
6, 6	143234	283941	144296	352664	7.05	169.602
8, 8	149734	444492	152298	682184	17.573	326.585
10, 10	154405	1238562	157914	2141686	31.489	261.727
12, 12	161549	1296843	167361	2275864	52.899	859.594

4. DISCUSSIONS AND PERFORMANCE STUDY

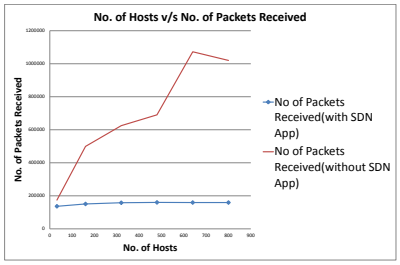
One can compare the performance of networks with and without the SDN App by varying the number of Hosts to study the amount of Network traffic as in Figure8a and Figure8b and the average RTT as in Figure8c. There is a huge gap in the number of packets received by the switches with SDN app setup and without in Figure8a. This huge gap is because the broadcast traffic is drastically reduced and the unknown unicast is completely eliminated as the flow tables have entries to forward the packet to any MAC address in the entire MAC address space with the help of the hierarchical pseudo MAC structure. ARP broadcast is reduced because there is only one ARP broadcast per one new IP and all ARP request are handled by the SDN app. On the other hand in traditional L2 learning setup, there is an ARP broadcast every time a host does not have an IP in its ARP table.

By observing the Figure8b, one can see that the transmitted packets increase exponentially with number of hosts. This is again due to broadcast traffic. When a packet is received by the switch with a broadcast MAC address, it floods that packet out all the ports except the source port. Consider a switch with 'n' ports which are connected to 'n' other switches. When this switch receives a broadcast packet, it will send the packet out of 'n-1' ports and those 'n-1' switches further repeat this process. Thus the number of packets transmitted grows exponentially with size of the network. Increase in the size of network implies increase in the number of ARP broadcasts and unknown unicasts. As in Figure8c, The average RTT with the SDN App is much faster and grows much slower with increase in number of hosts due to the following reasons:

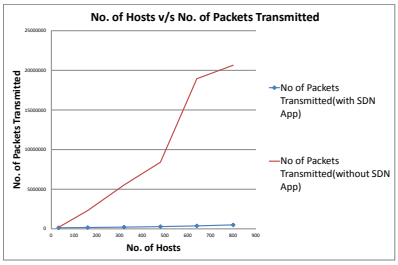
- Highly reduced broadcast traffic: The amount of traffic within the network is reduced drastically. The links are not choked by unnecessary broadcast packets and the associated queue delays are avoided.
- Compact flow table size: The number of entries in the switch flow table is very compact as the forwarding entries are hierarchical. With our application the size of the table does not grow as the number of switches remains same and only the number of hosts are varying. Smaller flow tables lead to faster lookups. On the other hand, the switches in traditional L2 networks have to maintain a flow table entry for each host. Hence the size of the table grows with the number of hosts and the lookup performance degrades with size.

DOI: 10.1002/nem

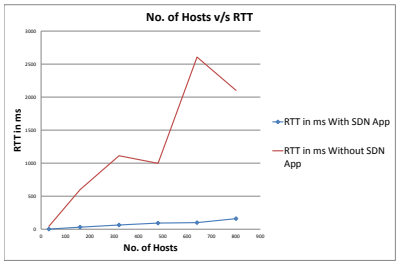
- Load balancing: In traditional L2 networks, the requirement of running a spanning tree protocol effectively disables the ability to simultaneously use the redundant links in the network. As a result, the links at the root of the spanning tree become oversubscribed - increasing the queue delay and becoming the performance bottleneck. With our architecture, the redundant links are effectively utilized to load balance the traffic and avoid clogging up of links, thus minimizing the queuing delays.
- One time forwarding insertion: With our architecture, the forwarding tables are computed initially based on the hierarchy and inserted only once. But in case of L2 forwarding, the flow insertion is reactive i.e., a new entry is needed every time a host is new with respect to the MAC table. Such entries have an associated timeout and will inevitably have to be inserted again and again. The overhead in terms of time taken for flow insertion is significant in comparison to forwarding of packets.
- Faster ARP responses: In traditional L2 network, the ARP request turnaround involves multiples hops. the request has to reach the target host and back again. With our app, the ARP requests are intercepted at edge switch and sent to the controller. In most cases, the ARP mapping is available at the controller and the ARP response is directly sent back to the edge switch, avoiding the overhead of multiple hops.



(a) No. of Hosts v/s Packets Received



(b) No. of Hosts v/s Packets Transmitted



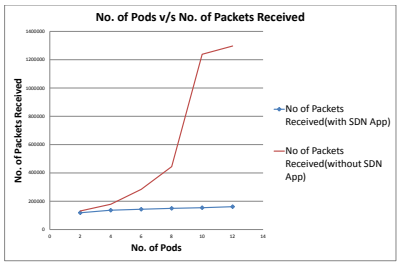
(c) No. of Hosts v/s RTT

Figure 8. Varying the Number of Hosts

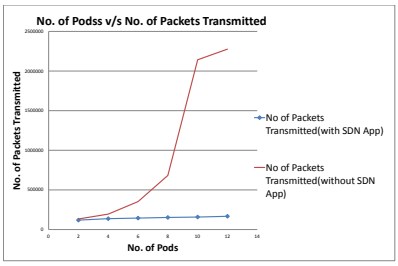
Studies were conducted by varying the size of the network. The number of core switches must increase in proportion to the number of pods. Otherwise there is a possibility that the network chokes near core switches. To achieve this, the experiments were repeated by varying the number of pods and the number of core switches. one can compare the results as in Figure9.

Prepared using nemauth.cls
DOI: 10.1002/nem

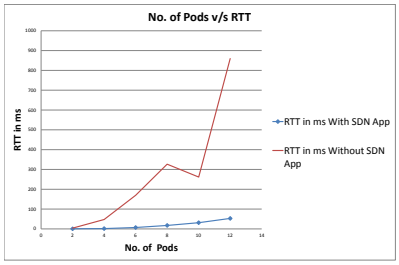
Here also one can do the same type of comparisons. When the size of the network increases there is a lot of performance gain. This strengthen our assumption that the performance boost is significant when we move to SDN based Data Center Network where we have still bigger network topologies.



(a) No. of Pods v/s Packets Received



(b) No. of Pods v/s Packets Transmitted



(c) No. of Pods v/s RTT

Figure 9. Varying the Numbar of Pods and Core Switches

5. RELATED WORK

For the implementation of SDN based 3-tier topology of Data Centers, Mininet emulator could be used. Mininet emulator has been proved as a good tool for the implementation of openflow based SDN networks. In [8] authors verified and compared the results of four real time projects with the Mininet emulator and found satisfactory results.

One of the initial efforts was using proxy ARP to reduce ARP Broadcast traffic [9]. One of the servers which knows the MAC address of the destination machine could give the reply to the requester as if the reply came from the actual destination. This reduces lot of traffic in the network provided the security issues are taken care properly. This can be implemented easily by maintaining an ARP table by the first hop switch like TOR switch which can generate the ARP reply as required by the requester.

It is highly desirable to implement large Data Center networks as L2 networks, and there has been a lot of research work done in this area. Seattle [4] uses the flat addressing scheme to reduce the scalability issues of L3 networks and grabs the simplicity of ethernet by implementing the networks as L2 networks. In Trill [10] protocol, data packets in the network can be forwarded using the Equal Cost MultiPaths (ECMP) rather than single optimal path as in traditional networks. This leverages

Prepared using nemauth.cls
DOI: 10.1002/nem

the built-in redundancy available in the links from core to aggregate switches and from aggregate to edge switches in Data Center networks. This helps in load balancing the traffic in Data Centers.

6. CONCLUSIONS AND FUTURE WORK

Internal traffic management in Data Center networks can be made simple with the introduction of Software Defined Networking (SDN) application and implementing the network as an L2 network rather than L3 network. It is shown that the performance of such Data Center networks can be improved with the introduction of the positional, hierarchical pseudo MAC addresses to reduce the broadcast traffic. Broadcast traffic due to unknown unicast is completely eliminated and the domain broadcast traffic is reduced significantly. By leveraging on the built-in redundancy available in the Data Center Networks, the dependency on the spanning tree protocol as in traditional L2 networks is completely eliminated. Our results shows that the broadcast traffic is highly reduced, the latency is reduced and as a result of that the throughput is increased. This application is well suited for emerging SDN based Data Center architectures. It can also be used for the migration of VMs in Data Centers.

In Future one can work on extending the current work to include multiple SDN Controllers which can share the information to manage the network in a distributed manner. The hierarchical addressing scheme can be modified to accommodate a scenario where multiple Data Centers might work in conjunction with each other. Network can be made more robust by considering link failures and switch failure situations.

ACKNOWLEDGEMENT

We would like to express our heartfelt thanks to PES University, Bangalore, India for providing the facilities to conduct the experiments related to this project.

REFERENCES

1. Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric, volume 39, number 4, pages 39–50 (2009). ACM SIGCOMM Computer Communication Review. ACM.
2. Bursky, Dave.Content-Addressable Memories Speed Up Network Traffic And More-Thanks to higher densities and new features, CAMs are taking on key roles in accelerating system performance. volume 48, number 2, pages 81–92 (2000). Electronic Design. New York, NY: Hayden Pub. Co.
3. Mohammad Al Fares, Alexander Loukissas, Amin Vahdat. A scalable, commodity data center network architecture, volume 38, number 4, pages 63–74 (2008). ACM SIGCOMM Computer Communication Review. ACM
4. Kim, Changhoon and Caesar, Matthew and Rexford, Jennifer : Floodless in seattle: a scalable ethernet architecture for large enterprises, volume 38, number 4, pages 3–14 (2008) ACM SIGCOMM Computer Communication Review, ACM
5. Md. Faizul Bari, Raouf Boutaba, Rafael Estevesy, Lisandro Zambenedetti Granvilley, Maxim Podlesny, Md. Golam Rabbani, Qi Zhang, Mohamed Faten Zhani: Data center network virtualization: A survey, volume 15, number 2, pages 909–928 (2013) Communications Surveys & Tutorials, IEEE
6. Veena S, Chandan Pal, Ram P. Rustagi, K.N.B.Murthy : Implementation of Simplified Custom Topology Framework in Mininet. In: 2nd IEEE International Conference, Asia Pacific Computer aided System Engineering-14, pp 0–6, ISBN 978-0-9924518 (2014).

Prepared using nemauth.cls
DOI: 10.1002/nem

7. Paulo Fonseca, Ricardo Bennesby, Edjard Mota and Alexandre Passito: A replication component for resilient openflow-based networking In: Network Operations and Management Symposium (NOMS), 2012 IEEE, pp.933–939, (2012)
8. Msahli. M., Pujolle. G., Serhrouchni. A. and Fadlallah. A.: Openflow and on demand networks In: Third International Conference, Network of the Future (NOF), pp 1–5, 2012, IEEE
9. Bitar, Nabil and Shah, Himanshu and Ghanwani, Anoop: ARP Broadcast Reduction for Large Data Centers : draft-shah-armd-arp-reduction-02.txt, (2011)
10. Perlman, Radia and Ghanwani, Anoop and Eastlake 3rd, Donald and Dutt, Dinesh and Gai, Silvano :Routing bridges (RBridges): Base protocol specification, (2011)
11. Lori MacVittie : What an Increase in East-West Traffic Patterns Really Means Technical report, November 18, 2013
12. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner: OpenFlow: Enabling Innovation in Campus Networks, Technical report, ACM SIGCOMM Computer Communication Review, Volume 38, Number 2, April 2008.
13. Munawar Hossain : Trends in Data Center Security: Part 1 Traffic Trends May 21, 2014 (cisco blog)
14. Jim Metzler and Steve Taylor : The great data center LAN debate Network World — June 8, 2011
15. Software Defined Networking, https://www.opennetworking.org/sdn-resources/sdn-definition.

DOI: 10.1002/nem