SSED: Servers under Software-Defined Network Architectures to Eliminate Discovery Messages

Emad Alasadi, and H. S. Al-Raweshidy, *Senior Member, IEEE*

***Abstract*— The high speed, low cost, sharing of peripheral devices and central administration features of the Ethernet have led to it being widely trusted as the backbone for recent networks. However, it suffers from many practical limitations leading to a lack of scalability, owing to its broadcast and multicast mechanisms, particularly in relation to the discovery processes. Whilst software-defined networks (SDN) have overcome many legacy network problems, scalability remains a major issue because broadcasting and multicasting have been inherited. Moreover, the problem is exacerbated with increasing network traffic, which results in higher bandwidth consumption, congestion and increased probability of a single point of failure.**

To address this, servers under software-defined network architectures to eliminate discovery messages (SSED) is designed in this paper and a backbone of floodless packets in an SDN LAN network is introduced. For SSED, flood discovery packets created by the dynamic host configuration protocol (DHCP) in the application layer and the address resolution protocol (ARP) in the data link layer are considered, respectively. SSED eliminates any broadcast discovery packets with better performance, lowers peak overhead and introduces an innovative mechanism for defining the relationship between the servers and SDN architecture. Experimental results after constructing and applying an authentic testbed verify that our proposed model has the ability to improve the scalability by removing broadcast packets from the data plane, reduction of control packets in the control plane, lessening peak overhead on the controller, preventing it experiencing failed requests, offering better response time and providing more efficient performance.

***Index Terms*— Software-Defined Networks, Scalability, Broadcast, Servers.**

I. Introduction

THE Ethernet is the most popular technology in local area networks that can be found in small geographic zones, such as in the home, on campuses and in enterprise network [1]. It allows for the sharing of resources with high performance, which supports virtualization principles and the client-server scheme in relation to the distribution of load among the servers as well as assisting in administration. The Ethernet protocol resides in the data link layer in the Internet protocol suite, providing services for its own layer and up layer protocols, such as broadcast ones like the Address Resolution Protocol (ARP) [2] in the data link layer and the Dynamic Host Configuration Protocol (DHCP) [3] in the application layer. It also services multicast protocols, such as the Bridge Protocol Data Units (BPDUs) [4], which is a multicast packet used by the Spanning Tree Protocol (STP) [4] in the link layer and the Multicast Listener Discovery (MLD) protocol [5] in the internet layer.

Despite broadcast and multicast protocols having the advantage of providing different services, such as getting destination MAC addresses, obtaining new IPs, loop free networking and discovering neighbouring nodes, the compulsory broadcast mechanism has resulted in multiple negative consequences that motivated us to design our model, which are as follows.

- As a consequence of broadcast packets, broadcast storms can happen in network topologies with multi-levels of connection, such as tree topology, causing further problems, such as congesting links, overloading the controller's/switch's CPU (we observed that experimentally the CPUs overloaded 100 % during approximately 0.5 sec of a storm) and generating MAC address flaps. The STP protocol in legacy networks is used to overcome the loop storm, however, it has the limitation of generating multicast traffic, which consumes bandwidth and supports only seven hops as a maximum bridged LAN diameter [6], thus restricting the network to scale. Practically, when increasing this to more than seven, the bridging loop problem takes down the whole the network as happened in [7], one of the worst IT crises in history.
- The broadcast mechanism leads to leaks in security, such as when the ARP protocol is used for different types of attacks, such as broadcast attacks, poisoning, spoofing and flooding, which result in the network being completely stopped or make resources unavailable, such as through a Denial of Service (DoS) flooding attack. In addition, sniffing can occur, whereby broadcast packets reach all of the hosts, even if they did not make a request, which can lead to data being intercepted by unauthorised hosts.
- Broadcasting leads to increased network traffic resulting in collision and competition at the same link, which leads to loss of packets, congestion and negative impact on response time. Hence, Cisco recommends in [8], which is a practical study, using no more than 500 devices in one

This work was supported and funded by Brunel University London.

Emad Alasadi is with the Department of Electronic and Computer Engineering, Brunel University London, U.K. (e-mails: emad.Alasadi@brunel.ac.uk).

H. S. Al-Raweshidy is with the Department of Electronic and Computer Engineering, College of Engineering, Design and Physical Sciences, Brunel University London, UB8 3PH, U.K., (e-mail: hamed.al-raweshidy@brunel.ac.uk).

collision domain. However, this limit to the Ethernet network means it cannot meet the needs of recent technology, such as the Internet of Things (IoT). In addition, the back-off algorithm used in collision networks to solve collisions, theoretically leads to the number of hosts being limited to 1,024 [9], which raises the scalability problem in these networks.

- The broadcast mechanism leads to increased CPU usage by the hosts, where practically hosts capture many broadcast packets which are irrelevant to them [10].
- All of the above limitations become worse, if the number of requests per second per workstation increases. In sum, the above issues have resulted in Ethernet being subject to lack of scalability [11], security leaks and limited reliability.

Another aspect that has motivated us regarding the proposal is that none of the extant available solutions has efficiently removed broadcast issues and scaled Ethernet. The techniques and their associated disadvantages are provided next, classified into two groups, according to the architecture of the different types of switches they use.

A. Related work with legacy switches

With this group, the proposed architectures use legacy switch architecture in their design to solve broadcast issues. In SEATTLE [12], the scalability is enhanced by using short path routing and a hash table, whilst no configuration (plug and play) is required, because it uses flat addressing. However, it adds another software program to the switch to perform some functions that are not supported by all types of switches, which leads to a lack of backward compatibility. Moreover, this switch requires an increase in the cache size when the number of hosts increases, for it has the responsibility of storing their additional information and also, it costs more than a traditional switch. In EtherProxy [13], a new device is introduced and inserted into the network, which partially stops broadcasting, but still causes a delay in response times owing to sniffing and analysis of each packet so as get its information. In addition, it uses distributed multiple EtherProxy devices that can find it difficult to synchronize the resolution tables among them. Moreover, a failing EtherProxy device can lead to isolation of all the switches and hosts behind it from the operation. In contrast, SAL [14] uses a distributed database in which the edge bridges store the host information. It does not completely eliminate the broadcast, because it still uses it among these edge bridges to retrieve the destination information and if a failure occurs in them all the switches and hosts will become isolated.

B. Related work with the SDN switches concept

These proposals use SDN concept architecture in their design to eliminate or reduce the effect of broadcasted protocols. In Portland [15], a dedicated centralised fabric manager device is introduced that contains information about the network. It adds another MAC format that is pseudo MAC (PMAC) to encode the hosts' position in the topology. The PMAC packet is forwarded to the Portland switch that contains special software for converting it back to actual MAC (AMAC), which increases the complexity of the network and the processing time in the switch. The other disadvantage of this method is that it cannot work with other switches, because the Portland switch has specific features, such as announcements to itself periodically. In Po-Wen, *et al.*'s framework [10] and the CPA framework [16] an ARP proxy in the Ethernet network is proposed as a module inside the SDN controller for handling ARP packets. It forwards every broadcast ARP request to the controller plane and generates an ARP reply message, which it sends back to the requested host. In addition, the former framework builds the DHCP function inside the SDN controller to deal with the DHCP broadcast packets. In FSDM [17], an ARP proxy and DHCP relay is introduced inside the SDN controller to handle ARP and DHCP broadcasted packets. The DHCP relay logically links the hosts and the ARP server, which leads to an increase in the number of packets the controller deals with, thus increasing the overhead on it.

For all the last three proposals (FSDM, CPA and the Po-Wen framework), proxy techniques are used inside the SDN controller, which has several disadvantages, such as lack of scalability in large networks at peak load due to increased request rates, resolution updates and mobility. This results in greater latency and response times, which get increasingly worse over time. In addition, there are controller overhead issues, fault-tolerance issues and single point of failure problems. Moreover, this increases the probability of attacks and as a consequence raises security issues, which is considered the most important problem nowadays, as reported by the SDN community [18].

There are effective features of the Ethernet protocol, such as self-configuration for serving the plug and play feature, centralised administration and the use of distributed servers, all of which should be retained when designing SDN-LAN architecture. However, when designing an SDN-LAN network the aim should be to eliminate unwanted features potentially inherited from legacy networks, such as the broadcast and multicast features that lead to increases in the number of hosts in one collision domain, minimise the number of protocols that are used in one domain over the Ethernet protocol, such as STP and increase the security as well as privacy for users. In addition, the SDN-LAN architecture design should possess some mandatory features so as to retain compatibility with legacy networks' hardware and software, such as keeping the stander protocols code without touch, thereby allowing for a gradual transfer from these networks to SDN. Moreover, no new architecture hardware should be added to the network as this would make it extremely difficult to change from the legacy architecture to the new SDN one. Finally, so as to allow for the continuing use of legacy switches no new software should be added to the host and switch side, otherwise there could be a backward compatibility problem among the hosts.

Hence, so as to satisfy all the aforementioned mandatory features, whilst solving the above Ethernet challenges, we propose the Servers under Software-defined network

architectures to Eliminate Discovery messages (SSED). SSED, firstly, introduces new mechanism for defining the relation between servers and the control plane in SDN architectures. Secondly, it handles broadcast and multicast messages that are generated by the most important type of broadcast protocols in the current Ethernet network paradigm. It eliminates all types of broadcast and multicast messages that could be generate by the broadcast or multicast protocols, such as ARP, DHCP, Network Time Protocol NTP [19], multicast STP BPDUs and multicast MLD protocol as well as any other future broadcast and multicast protocol, through the same SSED concept. In addition, it takes into account the peak load traffic and overhead issues.

The main contributions of this paper can be summarised as follows:

- It provides a practical solution for business purposes that has been experimentally proven to show that SSED is superior to legacy switches.
- A new method is proposed that defines the relation between the servers and the SDN architecture proactively and reactively by minimising the number of rules installed in the switches that are usually required in this kind of relation. For example, in a legacy network each host has its own rule in the switch, which leads to the number of rules increasing linearly with the number of hosts connected to that switch. However, using our model all the hosts share just one rule to deal with the server.
- Servers are used to offer different services concurrently, such as ARP and filter packets services in the SDN architecture, so as to decrease the overhead on the controller as well as for security purposes.
- A large authentic testbed with 23 computers is built and implemented to verify the superiority of the proposed SSED mechanism.
- A plan for eliminating the broadcast mechanism with load balance ability by using multiple servers in the same Ethernet network is proposed, which supports scalability.
- Reducing management packets in the control and data planes as well as minimizing the overhead on the controller at peak load.
- Protecting the controller from the effects of failed requests.
- SSED offers better response times and performance in the Ethernet network.

The rest for this paper is organised as follows. Firstly, in section II we describe the current method of broadcasting in legacy networks. In section III, description of the proposed SSED with its design and mechanisms concepts is provided. Then, in section IV, we introduce the implementations with flowcharts for SSED components. The testbed experiments are explained and the results presented in section V. Finally, section VI contains the conclusion and proposals for future work.

II. CURRENT SYSTEM DESCRIPTION

In this section, we describe the current broadcast Ethernet system by mathematically analyzing the broadcast phase using the learning switch mechanism and then explain the supported protocol.

A. Learning switch mechanism

Whilst the broadcast protocols are used for different services, such as obtaining destination MAC addresses or assigning new IP addresses for hosts, it uses the same broadcast mechanism. Generally, there are two different distribution phases to connect between source and destination hosts or servers. First, there is the broadcast phase from the source to request destination information or a service, and second, there is unicast, which is from the destination to the source to reply with destination information or in response to a requested service. There are some services, such as the DHCP containing more than two phases, for which every packet from source to destination before using the offering IP address is dealt with in a broadcast manner. The learning switch in SDN architecture it has same principles as a legacy switch. For the latter, see Fig.1(a), when the switch receives the broadcast packet for different types of broadcasted protocols, which is usually with the destination MAC address 'ff:ff:ff:ff:ff:ff ', it saves the source MAC address in MAC-to-port table so as to

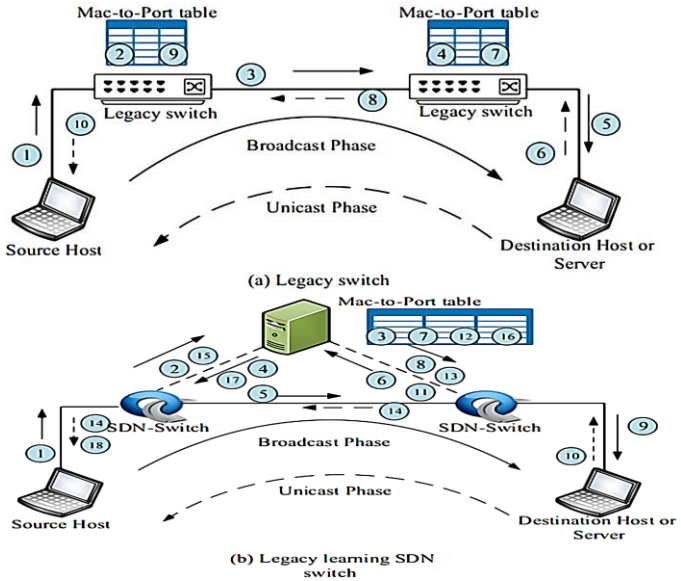


Fig. 1. Steps for filling the MAC-to-Port table and broadcast mechanism using legacy and SDN switches

prevent broadcasting subsequently, if the switch deals with same address again. Then it forwards the packet to all the other ports in that switch except that which inputted the packet. Then, the next switches use the same mechanism until the requested packet reaches the destination host or server. In case of the ARP service, the host that matches its IP address with the IP address field in the broadcast packet will respond with a unicast message that contains the IP and MAC addresses for the source and destination host. This unicast packet is, firstly, forwarded back to the switch that is connected to the destination host. After that, the switch saves the port and MAC addresses for the source and forward the

packet back in a unicast manner to the next switch that is already known, because it has already dealt with it during the first phase and so on. The learning switch in the SDN architecture can be seen in Fig.1 (b) and the difference is that the MAC-to-port table is stored in the SDN controller.

The number of packets in a learning switch network for a broadcast phase in the control plane and data plane in the SDN architecture depends on the number of switches (Ns) and the number of data links (Nl) connect to each switch. That is, to reach the destination host or server in broadcast mechanism the number of packets that are generated in control plane during the broadcast phase (Ncp) is a function of the number of switches F(Ns). The number of messages (Nm) between each switch and the controller is equal to two: one Packet_in message to the controller from the switch and one Packet_out from the controller to the switch to flood the packet to all the output ports, as in (1).

$$\text{Ncp during broadcast phase} = Nm * Ns \quad (1)$$

The number of packets that are generated in the data plane during the broadcast phase (Ndp) is a function of the number of switches and number of links F(Ns, NI), where one packet is excluded from each calculation, because it represents the input port and hence, is exempted from the flooding. Equation (2) represents approximately the number of packets generated in the data plane as a result of one requested broadcast packet from the host.

$$\text{Ndp during broadcast phase} = \sum_{1}^{Ns}(NI - 1) \quad (2)$$

B. STP and the learning switch mechanism

Practically, the learning switch forward mechanism needs support protocols that let it complete its work without a loop-network issue. Spanning-tree protocols such as STP, Rapid Spanning Tree Protocol (RSTP) [4] are used in spanning loop topologies to prevent a broadcast loop (loop-storm). A loop storm can happen between two switches that have multiple possible paths connecting them. In this case, the STP manages the network logically by ensuring the availability of just one possible path between two switches, which thus prevents a loop storm. The main disadvantage of STP is that it is a multicast protocol and to do its job it has to multicast BPDU packets among switches every 2 seconds [4], which puts more traffic in network and hence, causes delays in response time. In addition, in some cases the whole network can break down if it exceeds seven hops as the maximum bridged LAN diameter, which thus has to be taken into account when designing the network [7].

III. SSED DESIGN

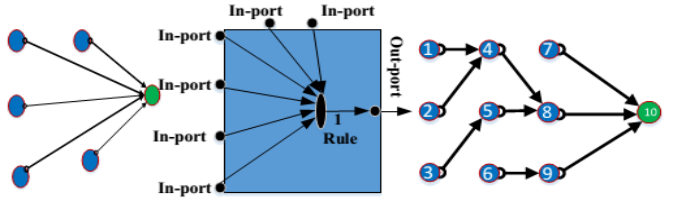
To deal with all the factors discussed in section I and to overcome the weaknesses in previous architectures, the SSED flexible framework has been designed by the introduction of Multi-To-One (MTO) collective service method and this is introduced first. SSED is used to define the relation between the SDN architecture and the servers proactively and reactively as well as eliminating different types of broadcast packet. The flexibility of SSED stems from its ability to forward packets to destinations chosen by the controller (not by the host), which depends on SDN controller management algorithms. This feature gives the SDN architecture flexible behaviours so as to be able to perform different functions, such as load balance, management and packet forwarding. Whilst the focus is on the ARP and DHCP broadcast messages, the same concept will work for all other broadcast messages.

A. Multi-To-One collective method

A new flexible collective service method that can be deployed in an SDN network is proposed, based on the ability to the controller to have a general view of this network. Its name comes from its job, which is defined as:

Directing multiple nodes that request the same service to a final node that offers the requested service using the unicast concept, in place of the usual broadcast request concept, with just one installed forwarding rule for each service.

Hence, it is called Multi point-To-One point or simply Multi-to-One (MTO). As shown in Fig.2 (a), MTO is a group communication and routing methodology, for which a set of nodes (or points) that needs a specific service is routed logically to a single node (or point) independently and in a unicast manner. By so doing, a message can be transmitted from any member in that set to the final single node, independently. The main features of the method are that the source nodes may not be related to each other, there is no limitation for number of nodes between the source nodes and the destination node and source node request service in a unicast manner, rather than through broadcast. MTO can be applied in different applications, for example, in a single SDN switch MTO connects logically multiple input ports from different sources to one output port using a single installed rule in the switch forwarding table, as shown in Fig.2 (b). The main advantage is that the switch can deal with N number of hosts without affecting the size of the forwarding table inside the switch, which leads to a decrease in its memory size and hence, lower cost.



(a) MTO general concept (b) MTO rule connects infinite number of In-port to one Out-port in SDN switch (c) MTO logically connects multiple source nodes to one destination node using unicast concept

Fig. 2. MTO concept with different applications in the SDN architecture

Fig.2 (c) shows another example for implementing MTO in multiple switch SDN architecture. Nodes (switches) 1,2,3,6 and 7 can connect logically to node (switch) 10 using MTO collective method with a unicast manner. The main advantage that is added when using the SSED architecture is that the controller can have a view of all the switches in its data plane by using the discovery mechanisms. So, it can forward the packet anywhere in its network, even if the message is not addressed to that destination and does this without modifying

the packet fields and as a consequence, this supports the client server concept.

B. SSED mechanisms

In the current Ethernet network, the broadcast mechanism plays an important role in performing different function, such as looking up MAC addresses associated with destination IP addresses by using the ARP protocol, assigning automatic IP addresses for hosts by using the DHCP protocol, solving duplicate IPs by using the ARP protocol and keeping time synchronous among hosts with NTP broadcast protocol. In this paper, the broadcast mechanism is replaced by the SSED mechanism, which makes the SDN controller and server share the responsibly of responding to reply messages depending on the type of service that is requested by the host. The server's role is to provide the service and the SDN controller's role is to ensure that it provides the best path to the source host and the resultant reduced management leads to lower overhead in its control plane.

By applying the MTO concept *inside* an SDN switch, SSED forwards packets that need the same service and come from different input ports to one output port. As a consequence, the number of rules that are installed in each switch to reach a specific server equates to one, regardless of the number of source hosts connected to that switch. In addition, the forwarding decision inside the SDN switch will depend on the layer 2 (packets with destination MAC equal to 'ff:ff:ff:ff:ff:ff') and layer 3 protocols that are supported by the switch to distinguish the type of service that is asked for by the host. For instance, the *arp_type* field needs to be able to distinguish an ARP packet and that the service that is needed is hence an ARP one. Moreover, if ports 67 and 68 are inside a UDP packet this can be used to distinguish it from a DHCP service and port 123 in such a packet can distinguish it from an NTP protocol [20]. By applying the MTO concept *outside* the SDN switch, the controller in SSED can manage flexibly the requested service from a host to any server that offers it with just one rule in each switch.

The SSED mechanism distributes packets according to packet type directly to a specific server using proactive installed rules, so that the number in data plane in the *Request* phase depends on the number of switches in the best path (Nsb) between the host and the server, while no packets enter into the control plane during this phase. During the *Reply* phase, the data plane also will depend on the Nsb, while in the control plane there is one Packet_in message from the server plus a number of Packet_out messages equal to the Nsb between the server and the requesting host. Equations (3) and (4) represent the number of packets generated in the data plane (Ndp) and control plane (Ncp), respectively, as a result of one requested packet from the host.

$$Ndp = 2 * Nsb \quad (3)$$

$$Ncp = 1 + Nsb \quad (4)$$

SSED contains some other mechanisms to perform its role, as follows.

1) Relationship between SDN controller and server's location

With SSED, the server's location is flexible in that it can be in the nearest switch to the controller as a data centre or can connect to any switch distributed in the same subnet. The network administrator can benefit from this flexibility in solving network issues, such as putting the server near to the switch that has heavy active users so as to reduce service access time [21]. In addition, the administrator can spread out the servers in the network to share the load among those offering the same service. This behaviour not only supports servers with broadcast protocols as it can also be used for any type of server in the same subnet.

2) SSED operation modes

In SSED, there are two types of behaviour that the controller uses to manage the connection between the servers and the hosts. Firstly, in bootstrap time, it uses the proactive mode to install rules in every switch so they can reach the Arp and DHCP servers (DHCP and ARP just as example; not limited to these) using the best paths available in that particular moment. If the network contains more than one server offering the same service, the controller can use distributed load algorithms, such as the Round Robin algorithm, to distribute the load amongst them. Secondly, if the network conditions change, such as a new server is added or removed, a switch is added or removed. Moreover, when there is congestion at the links then the reactive mode can be used to redistribute the load among servers or to change paths so as to be the best for reaching the servers.

3) SSED failure handling mode

Since there are different types of failure can be happened in an SDN network, SSED covers most of the important ones and gives solutions.

a. Handling Switch failure

The route between sources and destinations can be disrupted owing to failure in the routed switch or in its links. The controller in SSED uses a priority feature provided by OpenFlow protocol [22] to install two different rules in the same switch: a high priority rule for the normal route and a low one for failure mode, which can be used if a failure happens in neighbouring switches. In more detail, if a switch fails or the link to one goes down, then all the routes going through that switch will be disrupted. However, SSED deploys a new failure mode mechanism, whereby the controller identifies where the failure has happened by detecting the deleted port from a neighbouring switch and reporting a change in the port status. Then the controller reactively installs a new rule in the neighbouring switch or this switch will employ the second priority rule, proactively installed by the controller, in its forwarding table in order to keep the service offered by the servers working.

b. Handling Server failure

The controller, to do its management job properly, needs to create tables to use so it can track changes in the network. SSED creates a server-switch table in the controller containing the information about which switches are routed to which

servers. For example, switch1 is set to reach server X that offers service Y. So, if failure happens to server X, it is easy to know the switches that direct packets to it and hence, install a new rule in switch1 to redirect packets to another server that also offers service Y.

There is the possibility of a server failing performing any service in any network, not just an SDN, SSED uses an echo message that it sends periodically to each server in the server-information table that is stored in the controller to check if it still alive or not and accordingly, the controller updates the *status* field in this table as required. SSED, after it detects the failed server, will install new rules in the affected switches to route to another server that offers same service, if there is one. Otherwise, the controller forwards it to the default gateway so as to obtain the same service from servers in another network.

c. Handling Controller failure

There are several previously devised mechanisms that SSED can use to overcome this issue, such as back up with a different SDN controller [23] and using or changing the switches to standalone mode so as to take the responsibility for dealing with packets without an external controller [24].

4) Handling discovery (join, leave and mobility)

One of the most powerful aspects of SDN is its ability to discover its controlled network components in relation to cases of join, leave (normally or in a failure case) or mobility. SSED performs discovery with high accuracy and speed as well as dynamically detecting changes. There are two types of discovery in SSED a switch discovery mechanism and a host (could be a server too) discovery mechanism.

a. Switch discovery

SSED generates a switch-information table in the initial setup of the network and puts it in the SDN controller's memory. When a new SDN switch connects with the controller, there is an exchange of negotiate messages between the two, the information from which being used by the controller to register the switch in the switch-information table. Then, the controller starts sending discovery packets using the Link Layer Discovery Protocol (LLDP) [25] to discover the topology (how these switches are connected together) and registers that relation in the same switch-information table as pairs in the hash table. The important difference with SSED from other platforms is that it stops sending LLDP, because of the fact that any port or switch added to the SDN network must be reported to the controller [22]. This use of LLDP leads to a decrease in the excessive number of multicast packets that are usually generated [26]. In addition, if a switch leaves, then SSED performs the same procedure as for failure mode, as discussed above and then, removes the switch from the switch-information table.

b. Host discovery

SSED creates host-passport table when first establishing the network and puts it in the SDN controller's memory. If a new host joins the network, then there are two standard possible ways for setting up its IP, manually or dynamically, using the DHCP service. In both cases, static IP or dynamic IP, the host firstly must send an *ARP probe* packet, which is an ARP request packet with the sender IP address equal to all zeroes and the destination IP address equal to checked IP [27]. ARP probes are sent by the host in order to detect if there is a conflict IP with other hosts before commencing to use that IP. Then, the switch forwards that request packet to the SSED controller, which will register in the host-passport table all the requesting host's information, including the IP and MAC addresses provided by the host as well as the port number and switch ID resting with the switch. After this, the controller will forward the host's IP and MAC information to all ARP servers in the local network. This forwarding to all servers is an important step to load balancing among ARP servers that SSED is able to deploy. It is important to note that the host sends another type of packet, an 'ARP Announcement', which is an ARP request packet with the sender's IP address equal to the destination one, if no ARP conflicting reply message has been received as a result of the generated ARP probe packet [27]. An ARP Announcement is sent by the host to tell the other hosts that it is commencing to use the announced IP and the controller uses that packet to update the host-passport table with the valid IP address.

The host sends an ARP probe and announcement packet each time its IP changes. If the host leaves the network for any reason, such as failure or normal leaving and its departure is of no consequence, then no action is taken. However, SSED will wait for a set time of no activity for that host and will then delete it from the host-passport table, sending update messages to all the ARP servers instructing them to delete it too. The host-passport table contains a field with the name Last Activity Time (LAT) to record the time of the last activity by the host. Otherwise, if the host moves from one switch to another, it sends an announcement message to the controller that leads to the updating of the field *switch ID* just in host-passport table and the host can still use the same IP.

c. Server discovery

The controller, when the network is first established, creates a server-information table and any server joining the network sends an announcement message using the UDP protocol to define itself to the controller, which then inserts the server information in the server-information table. The server-information table is a hash table containing the MAC address, IP address (usually static IP), join date, leave date, status, pool range (used for DHCP), type of service, port number and switch ID fields. If a server leaves by planned leave, as stated in the *leave date* field in the server-information table, then prior to this at a set time, the controller will withdraw responsibility from that server and give it to another offering same service, such as ARP, DHCP, and NTP among others. For the controller to do this, the same procedure in proactive mode that we discussed above to install rules in selected switches and forwarding packets to the new server is used. However, if any server leaves unplanned, such as in a failure

situation, the controller, firstly, will remove it from the server-information table and will make another server available to provide its service (see subsection B.3.b in section III above). Finally, in the case of a move of a server from one switch to another, such as from one VM to another, the server will send an announcement message using the UDP protocol to define its new position and which switch it connects with. Then, the controller updates the *switch ID* field (which records the switch ID that is connected to that server) in the server-information table and server-switch table. The controller then updates the affected switches and instructs them to forward packets to the server's new location.

5) SSED without STP

As SSED stops all broadcast and multicast packets as well as taking responsibility for managing the forwarding of packets from sources to destinations, there is no possibility of a broadcast or multicast storm occurring in a loop network topology. Experimentally we stop STP from working in the SSED architecture in a loop network topology and we check the network to see that it is free from any storm. In contrast, stopping STP is impossible with legacy switches in a loop network topology.

6) Handling other issues related to broadcast

Broadcast packets can be used for different purposes even though they have not been actually designed for them, such as an attack on other components of network (controller, servers, hosts etc.) that lead to several security issues. In addition, broadcast packets can be used to solve duplicate IPs among hosts in same network, but these lead to high consumption of bandwidth and congestion at the links.

a. Handling security issues

Security matters have yet to be completely solved in legacy networks, as well as in SDN ones, as has been widely reported [19]. SSED handles security issues by dealing with each leak individually. Firstly, it does not allow any broadcast packet to reach the controller, thereby stopping any flood of ARP *request* or *reply* that is often used by hackers to attack it, stop its work or spoil its tables. In addition, SSED stops any broadcast packets among hosts, because all the packets it forwards are to a specific server depending on the service requested. This avoids ARP cache poisoning inside a host as can happen during a man-in-middle attack, one of the most common forms. Moreover, it results in the avoidance an Arp flood attack to a victim host that could stop it functioning or consume its resources. Furthermore, there is the absence of flooding packets that consume shared resources in network. SSED also does not allow any Arp reply packet to transfer through the network components except that containing in its IP field the IP for the ARP server as the source server. The IP server is completely transparent to users, which leads to increased security. Taking all these factors into account, these lead to reduced security overhead (high overhead generated as a result of checking each packet) on the controller, because the servers work as filters to it and just pass tested active packets. That is, the server will check the validation of a request then reply, whilst the controller simply undertakes management of it. For example, if host1 is an attacker and sends one or multiple ARP requests to attack the SDN controller, the ARP server will detect it is an invalid request by using a specific algorithm, and the server will not forward this request to the controller. As a consequence, there is reduced probability of an attack on the controller and reduced overhead in the control plane.

b. Handling duplicate IPs

Duplicate IPs occur when two clients on the same link use the same IP address concurrently, as a result, problems happen for one or both clients [27] such as over consumption of resources, flood storms and security issues. To solve this issue, usually in legacy networks the host, before using the IP address assigned to it by the DHCP server, broadcasts ARP Probe packets which is an ARP request with the 'sender IP address' field set to all zeroes and the 'destination IP address 'set to the IP address being probed [27]. The purpose of this is to check whether there is the same IP address available in the network so as to avoid duplicate IPs. To this end, SSED stops any such broadcast by immediately dropping the packet and the duplication problem is solved by a specific code in the controller. This is a major feature of the controller's management, especially when the network becomes big with many connected hosts. The controller checks the IP field in all the host-passport table rows in case of a new host joining the network or changing its IP address as a result of it awakening from sleep mode, changing the network interface from inactive to active mode and for other cases of change in connectivity [27]. Then, if there is duplication of IPs between at least two hosts, the controller sends an ARP conflicting reply message to at least one of the duplicated hosts, the choice depending on the registered times for that IP and the newer registered hosts will be chosen to change IP(s). Specifically, the conflict message notifies the user through a screen message to change the IP manually or to activate the sending of a DHCP discovery message to the DHCP server so as to obtain a new IP.

c. Handling Head-Of-Line blocking (HOL) phenomena

SSED deals with the head-of-line blocking (HOL) phenomena, which occurs owing to other packets are blocked when the packet at the front of the FIFO queue cannot move. The probability of its occurrence increases with the broadcast mechanism as more packets are generated on the output ports and so more compete to use them. SSED solves this, whereby the controller in SSED has a whole view on its network and installs different rules/paths to the ARP server in each switch with different priority levels. Hence, the first packet in FIFO is no longer waiting if the output port busy as the packet can go through different ports to reach its destination. In addition, SSED by eliminating the broadcast mechanism reduces the probability of HOL happening.

IV. SSED IMPLEMENTATION

We show in detail how SSED implementation handles ARP and DHCP broadcast messages by using a server-based

concept with that of MTO. We extend the RYU controller by adding the three SSED components as follows.

A. SSED bootstrap, proactive and reactive components

SSED combines proactive and reactive mechanisms, as can be seen in Fig. 3. Firstly, SSED-bootstrap starts with establishing the network by joining the SSED SDN controller, which directly starts to discover the network under its control. SSED establishes a switch-information table that contains a hash table (e.g. Source switch: [Destination switch, Source port, Destination port and Cost]), which will start being filled when a new switch joins the network. The SSED uses LLDP to discover the network and continues filling the switch-information table. The discovery of the switches process will be continued until a specific time as configured by the administrator so as to let all the available switches join the network. In addition, SSED creates a host-passport table that contains host discovery information and a server-information table that contains information about each server joining the network. After finishing the bootstrap time, both the switch-information table and the server-information table will be used in the SSED proactive-mode to find the best path between each SDN switch and the ARP and DHCP servers by using Dijkstra's algorithm, after some development. SSED proactively installs one forwarding rule for each server in each switch so as to let ARP and DHCP broadcast messages be forwarded directly to their respective servers without going through the controller. This mode uses the MTO concept as SSED completely ignores where the broadcast packets are coming from (i.e. from which sources) and just focuses on the output port (the rule is: do not care about input ports and go to specific output port). MTO will make one forwarding broadcast rule work with an infinite number of hosts to reach the ARP and DHCP servers that has no effect on switch memory. Proactive mode is repeated every threshold time to deal with any changes in the switches topology, whilst concurrently the ARP and DHCP components start working in the multi-thread concept. If a new switch joins the network, the SSED reactive mode adds it to the switch-information table and starts to discover how it connects to other switches.

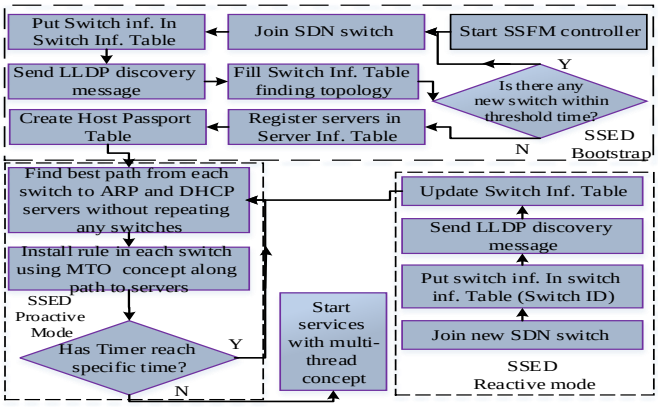


Fig. 3. A flowchart of the SSED bootstrap, proactive and reactive components

B. DHCP component

There are two ways to assign an IP address to a host, statically by using manual configuration or dynamically by using the DHCP protocol. With a dynamic IP address, the host sends broadcasted DHCPDISCOVER message to request an address from the DHCP server, which has pool of them to offer. The message will be entered into the nearest SDN switch, which connects to that host and the switch uses the MTO rule, which has already been installed in proactive mode in the bootstrap time, in all the switches along path to the DHCP server so as to forward that message. The DHCP server answers with a DHCPOFFER message, which is a unicast one that contains the host's MAC address in the target MAC address field in an Ethernet packet and this server's MAC address in the source MAC address field. This DHCPOFFER contains an offer of an IP from an IP pool in the DHCP server. The message will go back to the nearest connected SDN switch, which does not have a rule for forwarding and so it sends the message using the OpenFlow protocol as Packet-in to the SSED controller. SSED uses just one packet-in message to complete all stages of the requested service in order to eliminate overhead on the controller, especially during peak load. The controller catches the packet-in message and decapsulates it to get the DHCP information, subsequently checking the type of DHCP packet. Then, the controller checks whether it is a DHCPOFFER packet and the source MAC address to see that it belongs to one of the DHCP servers in the server-information table, for security reasons.

If *NO*, the controller will drop the packet, because it has come from an unauthorised source, if *YES*, the controller will look inside the host-passport table to update the existent host

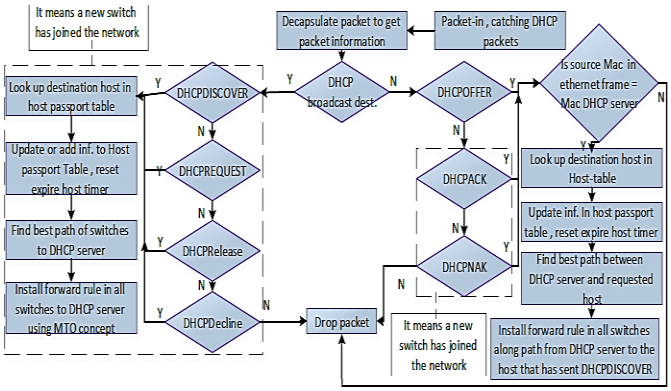


Fig. 4. Flowchart of the SSED DHCP component for handling DHCP messages architecture

record and reset the expire-host-timer field that is used to check whether the host is still alive. SSED uses the hash function to perform lookup in the host-passport table and considers the IP/MAC field in received packets as the key to finding the host's record. Following this, SSED controller finds the best path back to the host that has made the DHCPDISCOVER, installs rules to prevent next time Packet-in to the controller and then, forwards DHCPOFFER to the host. The host will generate a DHCPREQUEST message, which is also a broadcast message and sends it to the nearest switch that then forwards it to the server without notifying the controller with a packet-in, as the MTO rule already exists in the switch. The DHCP server will receive a DHCPREQUEST and then sends back a DHCPACK as a final agreement that

allows the host to use the requested IP address. The DHCPACK will be forwarded directly to the host that has made the request without notifying the controller as there is a rule already installed in that switch from the previous DHCPOFFER phase.

There are other types of DHCP messages that transfer between the DHCP server and host without send notification to controller in the SSED architecture. A DHCPNACK message is a unicast from the server to the host, letting it know that the requested IP address is not allowed owing to an error, such as it now being used by another host or it is no longer valid. In addition, there is DHCPRELEASE, which is broadcast message sent by the host to the DHCP server to let the server know it will log out from network. Moreover, DHCPDECLINE is a broadcast message from host to server to notify it there is an error in the configuration parameters. In exceptional cases, if DHCPDISCOVER, DHCPNACK, DHCPACK, DHCPREQUEST, DHCPDECLINE and DHCPRELEASE are sent as Packet-in to the SSED controller, this means that the switch sending the messages to the controller has just joined the network and so, the MTO rule has not yet been installed. Only a DHCPOFFER message should be sent in the Packet-in to the controller and just for the first time, because after that the server knows the route to the requesting host, unless there has been a change in the network topology. A detailed flowchart of the DHCP component in our model is shown in Fig. 4.

C. ARP component

The implemented Arp component contains two parts. Firstly, there is the ARP host discovery, which is described in detail in section III (host discovery mechanism). Secondly, the ARP service part refers to when a host needs to connect to other host it first having to get its MAC address so as to be able to send messages over the Ethernet. It sends an ARP broadcast REQUEST message to the nearest connected switch, which already proactively has had the MTO rule installed in it to forward any broadcast ARP REQUEST to the

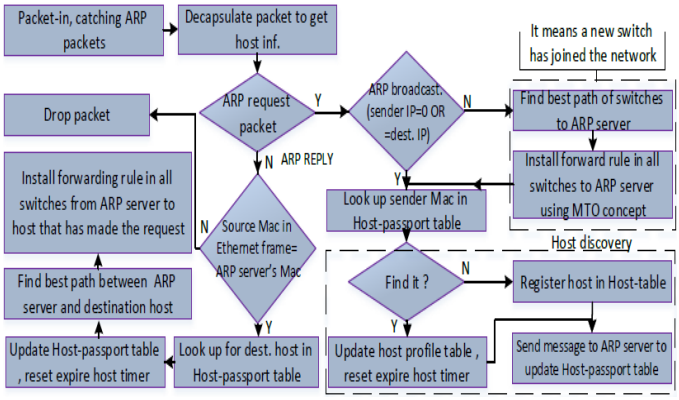


Fig. 5. Flowchart for handling ARP messages

ARP server. Consequently, there is just one rule to forward messages from infinite hosts to the ARP server following a unicast concept. The request is forwarded along the switches until it reaches the ARP server. The server then decapsulates the message and sends an ARP REPLY message with the MAC address for the destination host in the source MAC field, if it finds it in the host passport table, if does not then the server drops the packet. It is very important to let the server work as a filter just for valid requests so as to minimise the overhead on the controller. The switch that is connected to the ARP server will encapsulate ARP REPLY in a Packet-in message and sends it to the controller just the first time.

The ARP component in SSED will be triggered by the ARP packet and decapsulates it to find the type of ARP message and the destination host for it. If the message is a broadcast request message with a zero in the IP source field or the IP source is equal to the IP destination, then SSED deals with the packet as an advertisement message. If not, this means there is new switch joining the network and hence, the MTO rule has not yet been installed in it. So, SSED finds the best path between that switch and the ARP server and installs the MTO rule. However, if the message is ARP REPLY, then SSED checks the source field for that message and if it is not from the ARP server it then drops the message for security reasons. Otherwise, it looks up the host-passport table to update the host information and resets the expire timer, whilst also calculating the best path back from the ARP server to the host that has made the request. A detailed flowchart of the ARP component in our proposed algorithm is shown in Fig. 5.

V. TESTBED RESULTS

In this section, comprehensive testbed results are provided to demonstrate the performance of our SSED model. The testbed was built using 23 PCs, as can be seen in Fig. 6, twenty of which have the specifications of core 2 Quad, 2.66 GHz, 2.9 GiB memory and an Ubuntu 14.04 operation system. These can be used either as SDN switches by activating an open virtual switch (OVS) or as a host performing role of a single host or multiple-virtual hosts, depending on the experimental scenario.

Of the remaining three PCs, one works as a SDN controller, with the specifications of core i7, 3.40 GHz, 3.8 GiB memory and an Ubuntu 14.04 operation system. SSED uses a RYU SDN controller as the network operating system (NOS), which was developed by Nippon Telegraph and Telephone (NTT) as an open source operating system [28] that provides tools and libraries for design SDN components, which was written using the Python language for fast, easy and community supportive

Fig. 6. SDN Testbed environment with 23 computers

development. The SSED components are implemented under RYU using its expansive library. The final two computers are Samsung laptops, one of which works as the ARP server and the other as the DHCP server, both having the specifications of core i7, 2.20 GHz, 7.8 GiB memory and an Ubuntu 14.04 operating system. The SDN controller and OVS switches use

the OpenFlow protocol [22].

The experiments use one of two types of topology, linear or hybrid, depending on the purpose that they are designed to perform. Specifically, the linear topology is used in experiments that check the response time, because the effect element in this is the distance between the source and the destination, which is defined as the number of hops between them.

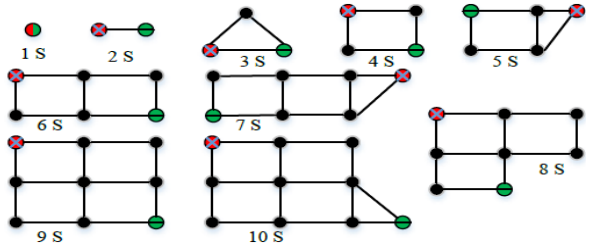


Fig. 7. 1-10 switches (S) in hybrid topology, the red circles with an (X) connect to the host and the green circles with a (-) connect to the server

The hybrid topology (it is used in real networks) is used in the experiments that are designed to evaluate the network traffic ratio (see Fig 7), because it is affected by the path that is chosen to reach the destination, which definitely is impacted upon by the mechanism that is used to forward packets.

To prove the efficient performance of SSED, several experimental scenarios were designed as follows. Note that the comparison involves the legacy switch scheme and our proposed scheme. There are two parts in this section, with the first dealing with traffic, whilst the second pertains response time.

The first part contains four experiments and concerns traffic in both the control and data planes. Ten PCs are used as OVS switches with a hybrid topology, as shown in Fig.7.

- The first experiment is performed to measure packet traffic in the control plane during 120-seconds bootstrap time and idle network behaviour using our SSED model and the legacy learning switch model.
- The second experiment is designed to measure the ratio of the network traffic to generation traffic, in both the control and data planes when generating 1 ARP of requested traffic under a traditional flooding scheme and our proposed scheme.
- The third experiment considers resource consumption for uncompleted requests, by calculating the ratio of network traffic to the generated failed requests in the control and data planes for both the SSED proposed scheme and the legacy scheme.
- The fourth experiment is performed to measure and compare the controller's CPU usage under the legacy broadcast scheme and SSED.

In second part, three experiments are performed to measure the response, latency and discovery time with 10 PCs being used as OVS switches in a linear topology.

- The fifth experiment is performed to evaluate the time for discovering host information on the server side and the latency in the controller when dealing with discovery packets. This is achieved by generating an ARP discovery packet from the host side and recording the receiving time for that packet on the server and controller side.
- The sixth experiment is run to measure the response time for receiving a service that is requested by a host.
- The seventh experiment is performed by increasing the number of hosts' requests per second on the SDN network. The aim is to evaluate the performance of the SSED during generated light, medium, heavy load from users working concurrently (how does increasing the number of requests affect the response time) on this network. It differs than previous experiment, in that it involves measuring how the response time is affected by sharing the network with multiple users. Then, the performance of SSED is compared with that for legacy schemes.

A. Bootstrap traffic: SSED and legacy scheme comparison

In this experiment, we first fix number of switches in testbed to 10 and use the hybrid topology in Fig. 7. In addition, no hosts are connected to the network so as to avoid traffic from them, with the focus thus being on traffic that is generated to establish the main parts of network, including the SDN switches and controllers. There are some processes start automatically during the bootstrap process without any external input (e.g. hosts), for example, the SSED switch discovery process and the legacy switch learning process. Thus, during the bootstrap time we can calculate how many packets are *in* and *out* from the control plane to establish the network before any host is connected to it. We run the experiment for 120 seconds, which is approximately enough bootstrap time for the discovery of 10 switches and the Wireshark tool is used to measure network traffic in the control plane. Gradually, over time, as can be seen in Fig. 8, the number of Packet_in and Packet_out in the control plane by using SSED is increased to reach 8,022 packets.

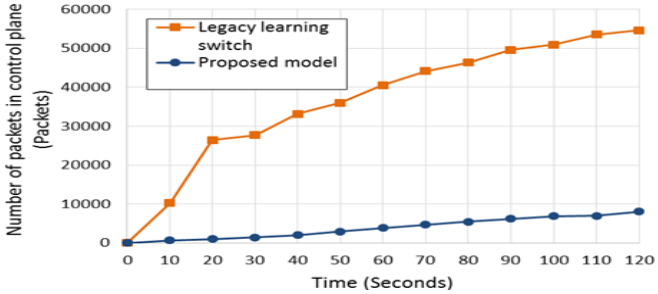


Fig. 8. Network traffic with SSED compared to the legacy learning switch mechanism during the bootstrap and idle network condition.

That number of packets is being generated because SSED during the bootstrap mode generates LLDP packets for management and switch discovery purposes. In addition, it drops all legacy management packets, such as a multicast listener report message in the MLD protocol, which is multicast by an IP node to report their interface status to their neighbours [5] and a multicast STP, which is used to build loop-free topology in a legacy network [4]. Regarding the traffic from using the legacy learning switch, this is significantly greater, rising to 54,653 packets, because it relies on flooding for discovery and on management services using MLD and STP. That is, when the time reaches 120 seconds,

SSED only makes 14.67 % of the overhead (number of packets) on the control plane that the legacy learning switch deploys for discovery and management services in the bootstrap period and when the network is idle. The idle network statistic is beneficial for evaluating the standard calculation that can help administrators find the threshold overhead on the control plane, thus potentially allowing for the determination of the hardware and software specifications needed for that plane.

B. Ratio of network traffic to generated traffic: SSED and legacy scheme comparison

In this experiment, we use hybrid topology and increase the number of switches from 1 to 10 switches. Only one ARP request message is generated from an edge host to the edge ARP server, as can be seen in Fig. 7, where the red circle with an (X) is the server and the green one with a (-) is the host. Then, by using the Wireshark tool the ratio of network traffic to generation traffic is calculated in both the control and data planes under the legacy flooding scheme and our proposed scheme. The main idea is send one ARP request packet and to measure how many packets will be generated in these two planes to get an ARP reply to the requesting host.

Regarding the control plane statistics, with an increase in the number of switches in the network the number of control message in control plane is approximately still the same or just slightly increases when using the SSED proposed model, because the ARP reply process needs just one Packet_in as the control message from the server to the controller plus a number of Packet_out messages equal to the number of hops for the best path between the server and the requesting host. For example, in Fig.9 it can be seen that no matter whether the network has 6, 7 and 8 switches, the number of control messages remains the same, i.e. five control messages, because there is one Packet_in to the controller from the server and four Packet_out to install the forwarding rule in the switches along the best path from the server to the host. With hybrid topology, sometimes the distance stays unchanged between the source (host) and the destination (server) when

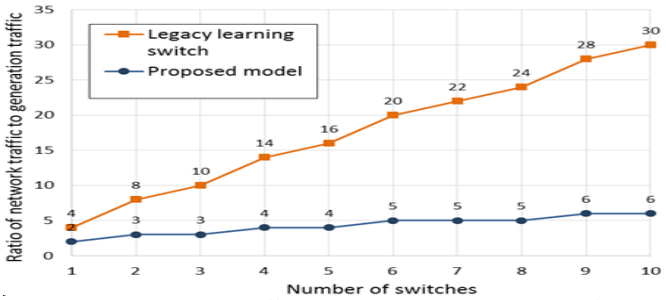


Fig. 9. Reduce network traffic in the control plane: comparison of SSED with and legacy learning switch mechanism

increasing the number of switches in the network, because there are a number of possible paths to connect these two entities, which is different to linear topology with only one possible path. On the other hand, for the legacy learning switch the network traffic increases significantly when the number of switches in the network becomes greater, because of the flooding of ARP broadcast packets to reach every switches in the network even though some are not on the path for reaching the destination.

The data plane statistics in Fig. 10, show that by generating one ARP request from an edge host with hybrid topology SSED keeps or uniformly increases (+4 packets per new switch) the ratio of packets to handle sending ARP reply packets by the ARP server to the host that has made the request (keeping or increasing the ratio depends on number of links that is needed to connect the host with the server). In contrast, the legacy learning switch increases the ratio practically linearly, because it floods the packet to every node in the network.

SSED needs just 44.44 % of the number of packets needed by the legacy learning switch to deal with 10 switches in order to send back an ARP reply to the sender because it benefits from the proactive installed rule in the switches using the MTO method to reach quickly the ARP server. In addition, it uses the Dijkstra algorithm to find the best path between the server and the host who has made the request.

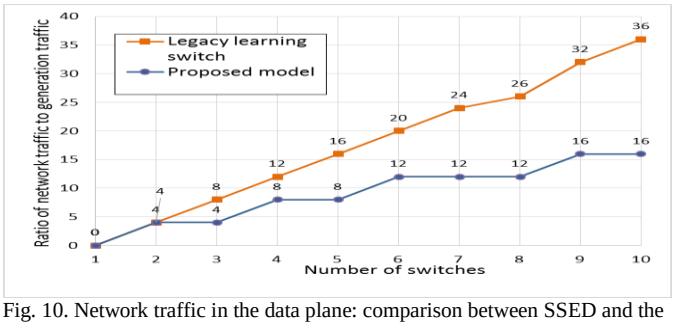


Fig. 10. Network traffic in the data plane: comparison between SSED and the legacy learning switch mechanism

C. Effect of retransmission traffic (resources consumption) on the control and data planes: comparison between SSED and legacy switches

To evaluate the effect of retransmitting traffic (that normally occurs in a daily network) with SSED, experiments for 10 hosts connected to an edge switch with 10 fixed SDN switches in a hybrid topology network is utilised. The hosts generate ARP faulty requests in order to obtain the goal of evaluating the effect of retransmitting traffic on resources consumption. A failed ARP request refers to not being able to find a requested destination's MAC address in the ARP hash table in the ARP server, which can be performed by sending requests to an unreached random destination. As a consequence of the failed request, the source starts retransmission of the ARP request multiple times as this is the normal behaviour of the Transmission Control Protocol (TCP) [29]. The main reason for retransmitting is that the source that made the request has not received a reply within a specific time period, which could be due to several causes, such as network congestion, interface error or buffer overflow. Other reasons for this are that the ARP server has not yet registered the destination host in its ARP hash table or the server has blocked its address owing to a security issue. Consequently, any ARP request asking for that a host's MAC address will not get a reply and this results in the retransmission of the same request from the source. To make the experiment

replicate daily used network traffic as closely as possible, different ARP request rates are generated from the 10 hosts at the same time.

Regarding control plane evaluation, Fig. 11 shows that with an increasing number of requests there is absolutely no effect on it when using SSED, because it proactively gives the responsibility to answer requests to the server that uses data plane for the purpose. The server works as a filter for reducing the number of requested packets and just passes the valid ones to the controller. This results in reduced overhead on the controller and eliminates the possibility of an attack on the controller. In contrast, the ratio of traffic using the legacy learning switch increases linearly to reach 24,000 packets of a Packet_in and Packet_out form for 4,000 requests in the control plane, because it floods every retransmitted request to everywhere in the network.

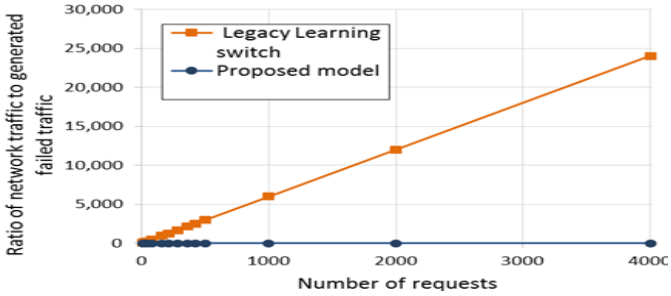


Fig. 11. Effect of the retransmission of traffic in the control plane: comparison between SSED and the legacy learning switch mechanism

In relation to the data plane evaluation (see Fig. 12), SSED generates 30% of the legacy learning switch traffic. That is, it provides a 70% reduction in consumption of network resources in the data plane when the number of failed requests reaches 4,000 with all hosts working concurrently.

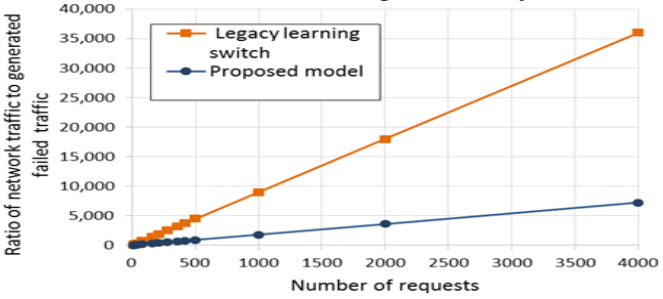


Fig. 12. Resource consumption during concurrently failed requests in the data plane: comparison between SSED and the legacy learning switch mechanism

D. CPU usage in the controller (SSED scalability)

In this experiment, a hybrid topology with 10 switches is connected and N number of virtual hosts are created and connected to the network. A fixed request number of a heavy user generation rate of 8 ARP requests per second is generated per virtual host with random IP addresses for sources and destinations. For this, the system monitor CPU tool in Linux is deployed to monitor the CPU usage, the measurement of which before any model being applied is 2.9% of core i7 CPU with 3.40 GHz, while it is 3.36% for SSED and 4.96 % for the legacy learning switch for the bootstrap communication management network.

As can be seen from Fig. 13, with a growth in the number of virtual hosts and a fixed rate of eight requests per second per host, the average percentage of CPU usage under SSED increases slightly from 6.31% to 12.5% for 1 to 500 virtual hosts (at peak load), respectively. It can clearly be seen that it reaches approximately a stable state after the connection of 50 virtual hosts concurrently owing to SSED's balanced multithread algorithm. This percentage of CPU usage is for handling ARP replies by finding the best path and installing rules in switches from the server to the host. However, when using the legacy learning switch the increase in (N) leads to an

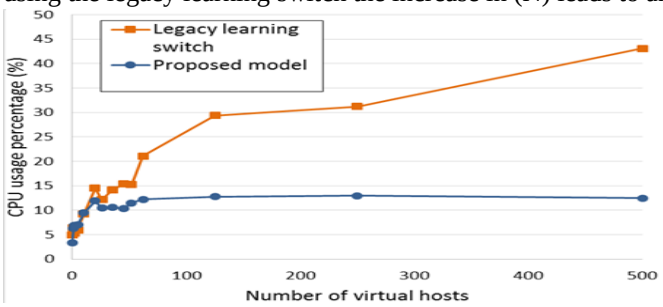


Fig.13 Average CPU usage in the controller for SSED and the legacy learning switch mechanism

increase CPU usage from 6.42 to 43.12 for 1 to 500 (peak load host number [8]) virtual hosts, respectively. This is because it has to handle many Packets-in and Packets-out per each ARP request owing to the flooding scheme as well as the complexity of the algorithm for finding the shortest path to the source. Next, the results of the second part of the experiments using the same constructed testbed, but with linear topology are reported regarding the three experiments relating time.

E. SSED host discovery time and controller latency

In this experiment, with the SSED model, linear topology is used with an increased number of switches from 1 to 10 and one host connects to one edge switch, while the ARP server connects to another. The host generates an ARP discovery packet, as explained in section III, which is entered as Packet_in to the controller, which then forwards this to the server. The latency time for the controller to complete the discovery packet forwarding process is evaluated and the results can be seen in Fig. 14. After that, the discovery packet will be received by the ARP server, which adds or updates the record in host passport table. The time that the discovery packet needs to reach the server from the requesting host is evaluated.

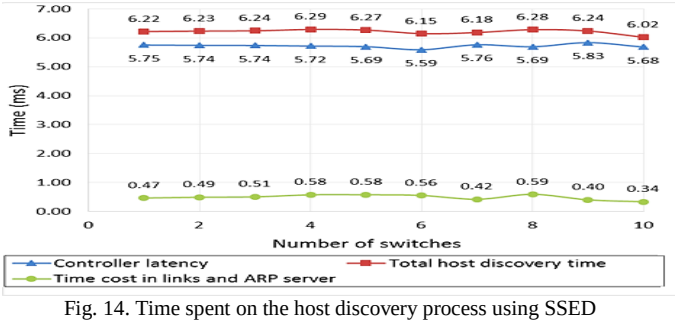


Fig. 14. Time spent on the host discovery process using SSED

It can be seen from Fig. 14 that the latency value and discovery host time have negligible impact on scalability, i.e. when the number of switches is increased, with the average

values for these being 5.71 ms and 6.21 ms, respectively.

This result is because SSED uses the controller as a link between the host and the server for the host discovery service, so the number of hops between the host and the sever is immaterial. In addition, the time that the host discovery packet spends at links and the ARP server is calculated by subtracting the total discovery time from the controller latency time, the average being 0.49 ms. This means that the discovery packet needs just 8.58 % of the time spent in the controller to pass through the links and server, which proves that the SDN controller owing to its complex nature (performing multiple jobs at the same time) spends more time compared with the distributed services that SSED uses to deal with broadcasted packets.

F. Response time

In this testbed experiment, a linear topology with an increase in the number of switches from 1 to 10 and generating one fixed ARP request using the ARPing tool from the host connected to an edge switch requests the MAC address for a destination host that connects to the network randomly. From Fig. 15, it is clear that with an increase in the number of switches, the ARP response time using SSED increases gradually at average of 0.19 ms for each added switch to the network.

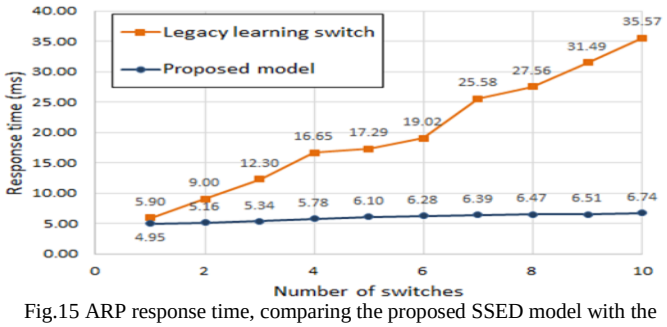


Fig.15 ARP response time, comparing the proposed SSED model with the legacy learning switch mechanism

This rate is as a consequence of sending a Packet_out message from the controller to the added switch to install the matching rule in order to match and forward the ARP reply from the ARP server to the host. By contrast, when using the legacy learning switch the growth rate is 3.29 ms on average, when adding a new switch to the path between source and destination hosts. This is because each added switch deals in the broadcast phase with one Packet_in by sending it to the controller as an ARP request and one Packet_out is sent by the controller to instruct the switch to flood that packet to all neighbouring nodes. After that, to handle the ARP reply on the way back from the destination host, the added switch sends one Packet_in to the controller to look up in the MAC-to-port table the source node that has made the request, which in turn, sends one Packet_out in the form of an ARP reply to the requesting host. As a consequence, as can be seen in Fig. 15, the response time practically linearly increases in proportion to the network switch scalability.

Regarding the scalability in relation to this experiment, by using the broadcast mechanism the ARP packet during the *Request* and *Reply* phases passing 10 switches requires 35.57 ms. Whilst SSED with that response time (i.e. 35.57 ms) can pass approximately 161 switches (whereas, as can be seen in Fig. 15, the response time for one switch is 4.95 ms and each added switch needs 0.19 ms). As a consequence, SSED scales the network approximately 94% more than the broadcast mechanism.

G. SSED Performance during different load

The performance and its stability for the proposed model is evaluated by generating light, medium, heavy and overloaded traffic from 10 concurrently working hosts. For this experiment, 10 fixed switches are connected with a linear topology. There are 10 hosts, each being connected to one SDN switch and the ARP server is connected to the fifth switch. The different rates of traffic sent concurrently from each host, are 1-4 requests per second (RPS) as light traffic users, 5-6 RPS as medium traffic users, 7-8 RPS heavy traffic users(at peak load [13]) and 10-12 RPS as overloaded traffic users. Each source host generates an ARP request for a MAC address for random destinations, each being designed to be unique in relation to all other requests from the same host so as to guarantee that they are not affected in any way by others requests.

As can be seen Fig. 16, with a light load traffic of 40 RPS as the total number of requests from 10 hosts working at the same time, the proposed model offers a better average response time than the legacy learning switch with the values being 16.34 ms and 23.86 ms, respectively. For a medium load with 6 RPS from each host, SSED also offers a better response time, the figures this time being 20.806 ms and 24.481 ms, respectively.

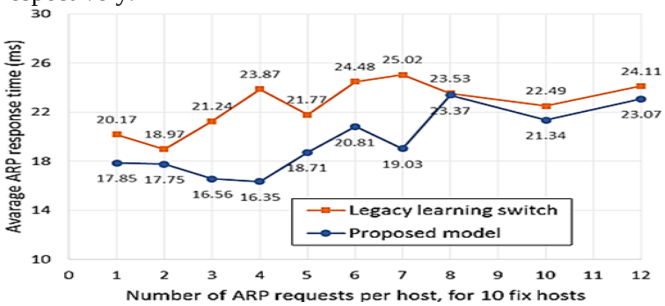


Fig. 16. The performance measures according to average ARP response time with different request rates from 10 fixed hosts connected to 10 switches

The same trend occurs for the heavy and overloaded scenarios, lead to the conclusion that SSED is efficient in terms of its performance as it well dealing effectively with increasing traffic rates. This is mainly because it handles ARP requests with fewer Packet_in and Packet_out than the legacy learning switch, which means less traffic is transmitted across the network and as a consequence, there is less competition as well as congestion at links, which in turn leads to lower response times. However, the average response time using the legacy learning switch increases with an increasing number requests, whereby each *request* entered to the switch will generate 1 Packet_in and 1 Packet_out until reaching the destination through all switches using the broadcast mechanism. Subsequently, each *reply* generates another 1 Packet_in and 1 Packet_out, if the switch was chosen as a hop within the shortest return path, otherwise (i.e if the switch is not chosen within the return path) the switch deals with just

the 1 Packet_in and 1 Packet_out that were generated during the broadcast phase. As can be seen from Fig.16, both approaches approximately meet at 8 RPS and there is 1 ms difference between them in 10-12 RPS, for two reasons. Firstly, there is the use of linear topology, which reduces the detrimental effect of the broadcast mechanism and hence, diminishes the response time when using that mechanism. Secondly, SSED, by using the ARP server at the middle switch, leads to more competition on that switch when increasing the requests and hence, increases the response time

VI. Conclusion and Future work

In this paper, firstly, the Ethernet network with its current switch features and the state of art architectures aimed at enhancing and overcoming its limitations have been discussed, Most of these drawbacks occur owing to the nature of usage of broadcast packets. To address these, the SSED architecture, design and implementation using several constructed testbed experiments with 23 computers to handle broadcasting packets was introduced, in particular, with the purpose of overcoming the side effects of broadcasting. The results have shown that the proposed model can eliminate broadcast packets from the network, thereby providing better performance. For future work, the aim is to implement SSED in the Internet with real traffic scenarios. In addition, the goal is to test it for all well-known broadcast protocols and services. Finally, the plan is to apply it to load balance among more than one server in a data centre network.

REFRENCES

[1] P. T. Congdon, P. Mohapatra, M. Farrens and V. Akella, "Simultaneously Reducing Latency and Power Consumption in OpenFlow Switches," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 1007-1020, June 2014.

[2] *An Ethernet address resolution protocol*, RFC 826, November 1982.

[3] *Dynamic Host Configuration Protocol*, RFC 2131, March 1997.

[4] *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*, IEEE Std 802.1D, 2004.

[5] *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*, RFC 3810, June 2004.

[6] *IEEE standards for local and metropolitan area networks: media access control (MAC) bridges*, IEEE Std 802.1D, 1990.

[7] S. Berinato, "All Systems Down", *CIO Magazine*, April 2003.

[8] J. Menga, "VLAN Operations" in *CCNP practical studies: Switching*, 1st ed, Cisco Press, 2003, ch. 2, sec. 3, p. 150.

[9] P. Dordal, "Ethernet," in An *Introduction to Computer Networks*, (1.8.22) ed, 2016, ch. 2, sec.2, p.48.

[10] P. W. Chi; Y. C. Huang; J. W. Guo; C. L. Lei, "Give me a broadcast-free network," *2014 IEEE Global Communications Conference*. IEEE, Austin, TEX, Dec.2014, pp. 1968-1973.

[11] T. Mizrahi; E. Saat; Y. Moses, "Timed Consistent Network Updates in Software-Defined Networks," *IEEE/ACM Transactions on Networking* , vol.PP, no.99, pp.1-14, Mar. 2016.

[12] C. Kim; M. Caesar; J. Rexford, "Floodless in SEATTLE: a scalable Ethernet architecture for large enterprises," in *Proc. ACM SIGCOMM*, Seattle, WA. , Aug. 2008.

[13] K. Elmeleegy and A. Cox, "Etherproxy: Scaling Ethernet by suppressing broadcast traffic," in *IEEE INFOCOM*, Rio de Janeiro, 2009, pp. 1584-1592.

[14] A. Shpiner; I. Keslassy; C. Arad; T. Mizrahi; Y. Revah, "SAL: Scaling data centers using Smart Address Learning," *10th International Conference on Network and Service Management (CNSM) and Workshop. IEEE*, Rio de Janeiro, Nov. 2014, pp. 248-253.

[15] R. Niranjan Mysore *et al*., "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM*, Barcelona, Aug. 2009.

[16] H. Cho; S. Kang; Y. Lee, "Centralized ARP proxy server over SDN controller to cut down ARP broadcast in large-scale data center networks," *2015 International Conference on Information Networking (ICOIN)*, Cambodia, 2015, pp. 301-306.

[17] J. Wang, T. Huang, J. Liu and Y. Liu, "A novel floodless service discovery mechanism designed for Software-Defined Networking," in *China Communications*, vol. 11, no. 2, pp. 12-25, Feb 2014.

[18] D. Jorm, "SDN and Security," *Open Network Operating System (ONOS).* [Online] 3rd April 2015. Available from http://onosproject.org/2015/04/03/sdn-and-security-david-jorm/ [Accessed 15/09/16]

[19] *Network Time Protocol (Version 3) specification, implementation and Analysis*, RFC 1305, March 1992.

[20] J. Touch *et al*., "Service Name and Transport Protocol Port Number Registry," *The Internet Assigned Numbers Authority (IANA)*, Aug 2016.

[21] M. Bienkowski, A. Feldmann, J. Grassler, G. Schaffrath and S. Schmid, "The Wide-Area Virtual Service Migration Problem: A Competitive Analysis Approach," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 165-178, Feb. 2014.

[22] *Openflow switch specification*, version 1.5.0 (Protocol version 0x06), Dec 2014.

[23] G. Yao; J. Bi ; L. Guo, "On the cascading failures of multi-controllers in Software Defined Networks," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Goettingen, pp. 1-2.

[24] *Open vSwitch manual*, 2nd ed., OVS, 2014, p.10.

[25] *IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery*, IEEE Std 802.1AB, 2016.

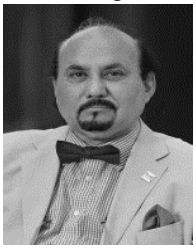
[26] U. C. Kozat; G. Liang; K. Kökten; J. Tapolcai, "On Optimal Topology Verification and Failure Localization for Software Defined Networks," *IEEE/ACM Transactions on Networking* , vol.PP, no.99, pp.1-1, Nov. 2015.

[27] *IPv4 Address Conflict Detection*, RFC 5227, July 2008.

[28] R. Kubo; T. Fujita; Y. Agawa; H. Suzuki, "Ryu SDN Framework:Open-source SDN Platform Software," *NTT Technical Review*, vol. 12, no. 8, Aug. 2014.

[29] *Transmission Control Protocol Darpa Internet program protocol specification*, RFC 793, September 1981.

Emad Alasadi received the B.Sc. degree in computer and software engineering from Al-Mustansiriyah University, Baghdad, Iraq, in 2003 and the M.Sc. degree in computer engineering and information technology from University of Technology, Baghdad, Iraq, in 2006. He is currently pursuing the Ph.D. degree in Electronic and Computer Engineering at Brunel University, London. His research interests include software-defined networking, network architecture.



Hamed Al-Raweshidy (SM'03) is Professor of Communications Engineering and received his BEng and MSc from University of Technology, Baghdad in 1977 and 1980 respectively. He received his PhD in 1991 from Strathclyde University in Glasgow, UK. He currently is the Director of The Wireless Networks and Communications Centre (WNCC) at Brunel Uni-versity London, UK. He has published over 380 papers in International Journals and referred conferences.