

An SDN Approach to ARP Broadcast Reduction in an Unstructured Campus Network

¹A.M. Ammu, ²Reshmi B Nair, ³P. Lolitha and ⁴Nima S Nair

¹Department of Computer Science & IT,
School of Arts & Sciences, Amrita University Kochi.
ammuum10@gmail.com

²Department of Computer Science & IT,
School of Arts & Sciences, Amrita University Kochi.
reshmi20dec@gmail.com

³Department of Computer Science & IT,
School of Arts & Sciences, Amrita University Kochi.
lolitha145@gmail.com

⁴Department of Computer Science & IT,
School of Arts & Sciences, Amrita University Kochi.
nimasnair@gmail.com

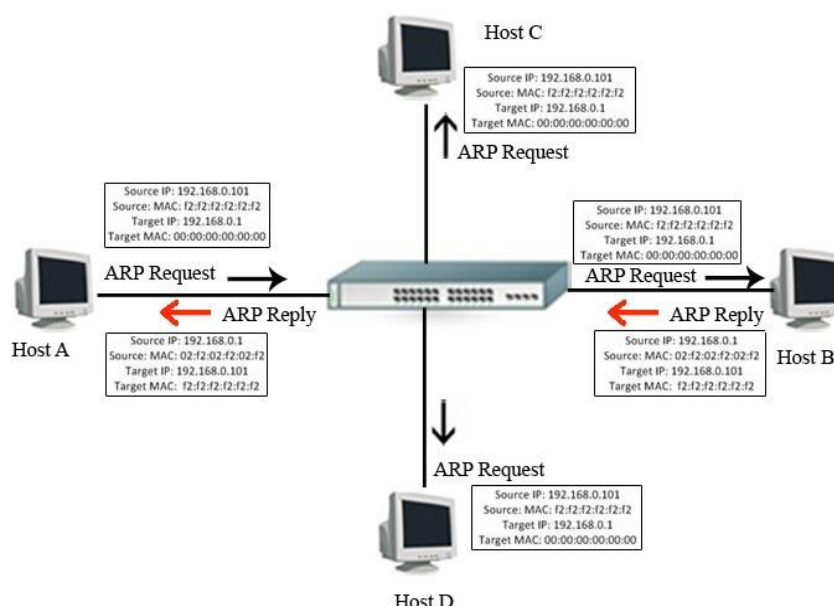
Abstract

The address resolution protocol is essential for communication between computers in a local area network by mapping a network address to a physical MAC address. A single Ethernet network cannot scale to larger networks such as enterprise or campus networks because of broadcast request from the network machines. This ARP broadcasts consumes a major portion of network bandwidth. SDN is an advancing technology, which isolates the two major functionalities in the network devices like switches and routers. This provides programmability and thus centralized control over the entire campus network. In particular we propose a general approach to handle ARP in SDN that is applicable in several network scenarios and is transparent for existing devices. In this paper we programmed the SDN controller to keep surrogate ARP table. All the ARP requests are then routed to the controller and it resolves the request. Proposed system is then compared with the standard methods and proved the efficiency by enhancing scalability of Ethernet by reducing broadcast traffic.

Key Words: ARP, broadcast reduction, SDN, surrogate ARP table.

1. Introduction

ARP is the protocol used to map the IPv4 address to physical address over Ethernet. When a source device wants to communicate with another device, source will check the ARP table in the cache for resolved MAC address of destination device. If it is there, it will communicate using that address. If ARP resolution is not there in the table, the source will broadcast an ARP request to the local network which contains sender MAC address, Sender IP address, and Target MAC address and Target IP address. The message is received by each device on the local Area Network and each device compare the Target IP address with its own IP address. When there is a match, targeted device will send back ARP reply message [1][2]. The destination and source device updates the ARP table for further communication. Fig.1 explains the ARP process in detail.



We are considering an unstructured campus network for our study. The Unstructured networks may not have a predefined structure and specifications. They are not customized for the operations of the network. On the other hand networks of large organizations and data centers have their own custom build networks and infrastructure. In unstructured campus networks the bandwidth consumed for ARP packets may slow down the network. The number of hosts in the network is directly propositional to the volume of ARP broadcast in the network. As the network scale up the ARP broadcast packets in the network also increases. It consumes network bandwidth extensively. We introduced the concept of SDN to take care of this scenario [3].

SDN is a new spurring network framework, which separates the control plane and data plane of forwarding devices in the network. The layer 2 switches and

routers are normally performing routing and forwarding functions. These devices check the forwarding/routing table to identify the outgoing path and forwards frames/packets to that port. But in SDN the functionality of calculating the route is taken from the devices and placed it in a device/system called controller [4]. The network devices only forward the packets according to the entries in the Flow Table. It consists of forwarding rules, which will then matched against the incoming packet header. The matched packets are routed to the destination through the specified port. Rest of the flows is forwarded to the controller. The controller then decides what to do with the packet and adds the rule to the Flow Table [5]. The communication between the OpenFlow enabled switch and the controller is specified in OpenFlow Protocol. As the controller is dedicated for handling packets, we can also program the controller for additional functionalities and handling forwarding devices. This programmability adds flexibility, agility, and transparency to network functionality. The network administrator can also add different type of devices and manage them irrespective of the manufacturer [6].

The main components of SDN are Controller, Northbound API and Southbound API. Controller is the central component which provides central view of entire network; manage flow control to the networking devices through Southbound APIs and applications and services through Northbound APIs. Southbound APIs are used to transfer information to switches and routers. Northbound APIs transfer information to applications and business logic [7].

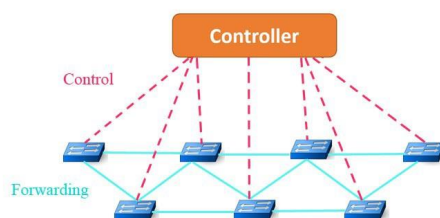


Figure 1.1

In our paper we are using these features of SDN in combination with ARP to limit the number of broadcast messages over Ethernet. We are proposing a Surrogate ARP module implemented inside the controller, which precisely processes all ARP requests. The need for ARP broadcast is then eliminated.

The paper is organized as five sections. Section 2 deals with related work done in the area of broadcast reduction. In section 3 the proposed architecture is described in detail. The implementation and comparison with the standard network is described in section 4. The paper is concluded with section 5.

2. Literature Survey

Many attempts have been already done by researchers to cut down the ARP traffic. All these have tested and implemented in normal IP network.

EtherProxy [8] is an approach introduced by Khaled and Alan Cox, in which they tried implement a caching ARP (IP – MAC Pair) in switches. This EtherProxies will back by the requested MAC-ID if the stash had the MAC address. This approach will work only in the area covered by each switch. L2 Population mechanism is another approach under Openstack, to distribute cache in the hosts [9]. SAL [10] devised a selective mechanism using distributed database. The main problem of L2 and SAL is the distributed database design and in case of SAL, the network scalability is obstructed.

Instead of this idea of distributed database, VL2 [11] and PortLand [12] use a centralized database method. These both approaches require location-based addresses which are locator-specific address and Pseudo MAC (PMAC) respectively. To transfer VMs from one realm to another requires new IPs and network reconfiguration. A cache on Forwarding Table (CFT) method has been devised by Xiaocui Sun and Zhijun Wang for Metro Ethernet [13]. By attaching IP address to the forwarding table they enabled the caching effect in forwarding tables of switches in the metro network.

A model of a Centralized ARP Proxy Server [14] is introduced by Hyunjeong Cho for large scale datacenter networks. The central ARP Proxy here maintains the IP – MAC mapping of the virtual machines and physical machines in a datacenter network. This approach can efficiently process ARP by reducing the broadcast traffic.

3. Proposed Frame Work

In this paper we proposed an innovative approach to eliminate the need for ARP broadcast in the unstructured campus network. The SDN controller is configured to keep a surrogate ARP table. This module is responsible for mapping the IP address to corresponding MAC ID. Since the SDN controller has a centralized view of the network it is easy to manage. The proposed frame work is given in the figure-3.1 given below.

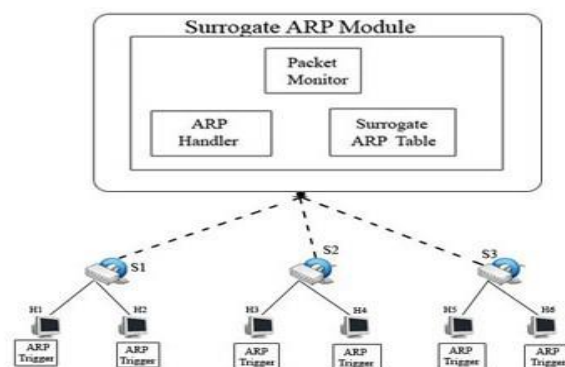


Figure 3.1

The entire process is divided in to two parts. First part involves an ARP Trigger module which installed in the host machines in the network, which is responsible for populating the Surrogate ARP table in the SDN controller. The second module is the Surrogate ARP module, which is implemented in the SDN Controller. The functionality of this module is to verify the ARP packet and perform the IP to MAC conversion. For this operation we implemented three functional components namely ARP Handler, Packet Monitoring Module and the Surrogate Table itself.

Initially the Surrogate Table in the SDN controller has no entries. It will be populated when the machines in the network starts. After booting and obtaining IP address from the corresponding DHCP servers in the network, the ARP trigger module is invoked. It will send a packet consists of IP address and MAC ID to the Surrogate module in the Controller. Thus when the hosts in the network come online the IP and MAC ID will be transferred to the central table. The ARP handler in the controller is always listening for this trigger packet. When it receives the trigger packet it will check the Surrogate table indexed by the MAC ID. The MAC ID of this packet is then compared with the table entries. If an entry is there, it will then check the corresponding IP address. If the IP matches it will leave the trigger packet. Otherwise it will modify the IP address with the new one. If there is no entry in the table for the MAC ID, the ARP Handler will add a fresh entry to the table. This process is necessary to remove inconsistencies in the ARP Table, as the IPs are assigned to the hosts by DHCP servers. The hosts can receive different IPs at different time and the table may contain old entries.

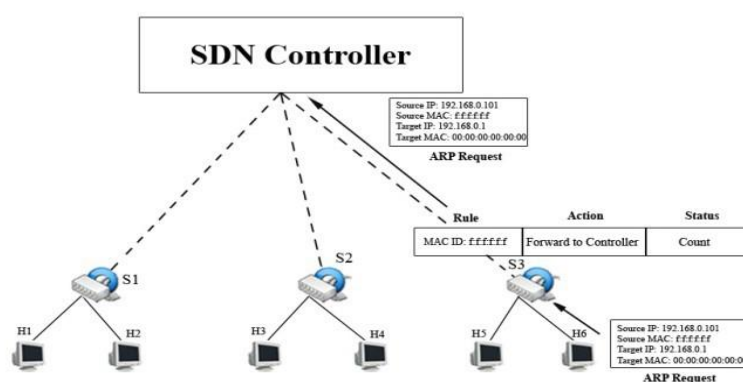


Figure 3.2

Whenever an ARP broadcast request is issued by the hosts in the network, it first reaches the neighboring OpenFlow enabled switch. The switch is preconfigured and it will not allow any broadcast packets to pass through it. A rule is set in the forwarding table to redirect broadcast packet to the controller. The packet

will be then forwarded to the controller for taking forwarding decision. Figure 3.2 shows the working. The Packet Monitoring program in the Surrogate Module will identifies the ARP Packet. The request packet consists of Source IP, Dest. IP, Source MAC ID and Dest.MAC ID as 00-00-00-00-00-00. The Packet Monitor then extracts the Dest. IP from the packet and invokes the ARP handler by passing Dest. IP as an argument. The Handler will search the table by IP Address for a match and returns with the Dest. MAC-ID. The Packet Monitor reconstructs the ARP reply packet and returns it back to the switch. The packet is then routed to the sender as usual ARP reply packet. This process phase out the need for broadcast and saves the network bandwidth substantially.

4. Implementation

We implemented and tested the framework in Mininet network simulator that supports SDN and OpenFlow. An unstructured network (LAN) topology containing three open flow switches and limited number of hosts is created in Mininet for implementing our work. The Mininet topology is given in the figure 4.1 given below.

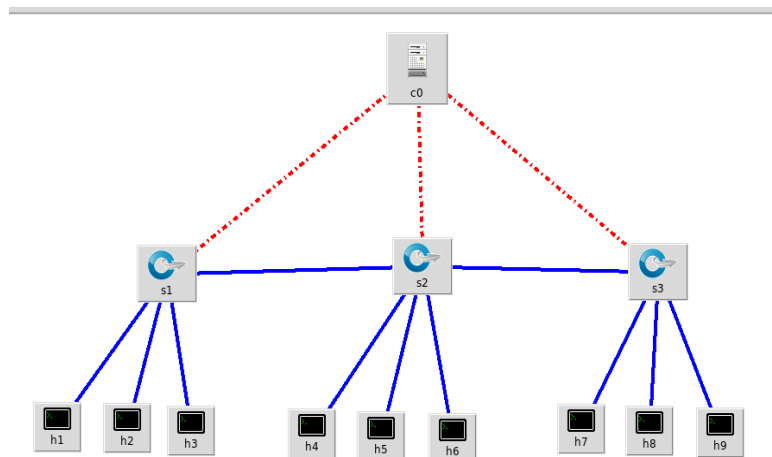


Figure 4.1

The ARP Trigger Module is implemented and installed in every hosts in the network using Python. The algorithm for the function is as follows.

ARP_Trig()

1. Fetch_IP_MAC() → Collects IP and MAC ID of the machine
2. Build_Pkt(IP, MAC-ID) → Constructs a packet with IP and MAC-ID
3. Send_Pkt() → Sendd the packet to the controller (ARP Handler)

We use the default POX controller as our open source SDN Controller. There are three functional modules in the controller for handling the ARP requests.

ARP_Handler()

1. Trig_Listener_Thread() –Always listens in a particular port for ARP requests and ARP Triggers

If Receive_Pkt(IP,MAC-ID) from ARP_Trig() then check the consistency of the table

- a. If consistent () then return 0
Else call Refresh_Table()
 - b. If (Trig_IP != ARP_IP) for all entries in the ARP Table
Insert (IP, MAC_ID) to ARP Table
2. Packet_Monitor()
 - a. Receive_Pkt() (Receives packet from the switch)
 - b. If receive_Pkt=ARP_Pkt extract Dest_IP call ARP_Handler(Dest_IP)
 - i. If (Dest_IP matches to ARP_IP) return Dest_MAC_ID
 - c. Build_ARP_Reply(Dest_MAC_ID, Dest_IP) and pass to Source.

In a normal IP based network the volume of ARP packet is shown below in figure 4.2. Nearly 15 – 20 % of the traffic contains ARP packets. Using our method we can completely eliminate the traffic with the concept of Surrogate table.

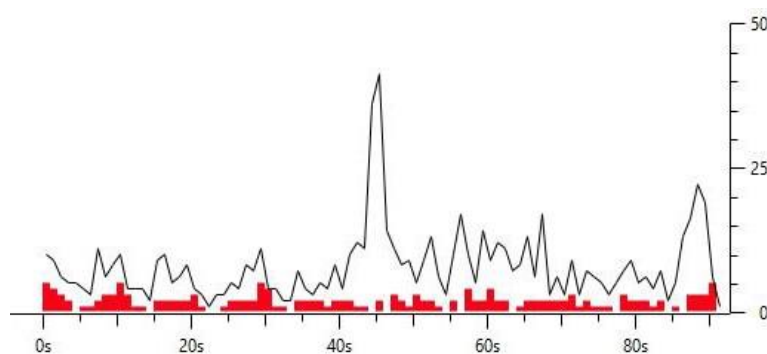


Figure 4.2

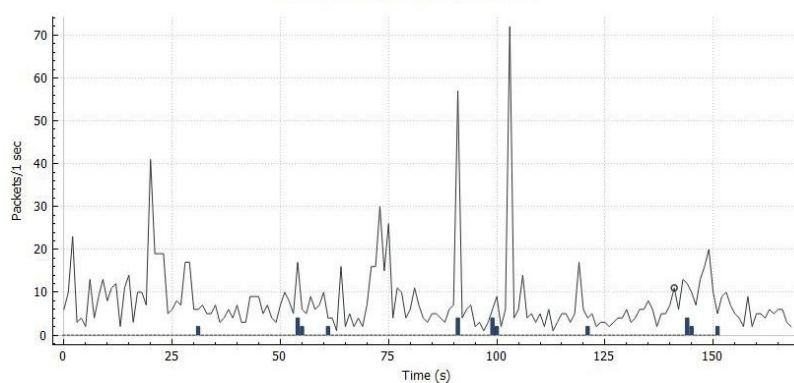


Figure 4.3

In our unstructured network after implementing SDN and Surrogate ARP module we can actually remove the ARP packets from the network. Figure

4.3 the graph shows the traffic that contains no ARP packets. The graphs are generated using Wireshark Packet analyzer by capturing packets in the network. Thus our work is proved to be efficient and can be used to remove broadcast ARP traffic in an unstructured network.

5. Conclusion

ARP broadcasts in unstructured networks slowdowns the network functionality when number of nodes increases. Our objective is to remove the ARP broadcast traffic in an unstructured campus network using SDN concept. An SDN network is centrally controlled, programmable and agile in nature. We implemented a surrogate ARP module in SDN controller and programmed the controller to keep the IP-MAC pair information of entire network, manage the ARP table and process the ARP requests from hosts. This method wipes out ARP broadcast packets from the network and all requests are handled by the central controller. We implemented this approach and proved to efficient. In our future work we concentrate on better methods to implement the Surrogate ARP table to reduce the search time in the controller.

References

- [1] Andris Sidorovs, Guntis Barzdins, Janis Lacis, Karlis Ogsts, SmartARP: merging IP and MAC addressing for low-cost gigabit Ethernet networks, *Computer Networks* 31 (1999), 2193–2204,
- [2] Mohamed G. Gouda, Chin-Tser Huang, A secure address resolution protocol, *Computer Networks* 41 (2003), 57–71.
- [3] C. Carter, Seung Yi, R. Kravets, ARP considered harmful: manycast transactions in ad hoc networks, *IEEE Wireless Communications and Networking* (2003).
- [4] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen , Katia Obraczka, Thierry Turletti, A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, *IEEE Communications Surveys & Tutorials* 16 (3), 1617–1634
- [5] Tothoonchian A., Gorbonov S., Ganjali Y., Casado M., Sherwood R., on controller performance in software-defined networks, *USENIX workshop on Hot Topics in management of Internet, Cloud and Enterprise Networks and Services (Hot-ICE)* 54 (2012).
- [6] Wenjuan Li, Weizhi Meng, Lam For Kwok, Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices, *Future Internet* 6 (2014), 302-336.
- [7] Akram Hakiri , Aniruddha Gokhale , Pascal Berthou , Douglas C. Schmidt, Thierry Gayraud, *Software-Defined Networking:*

- Challenges and research opportunities for Future Internet, *Computer Networks* 75 (2014), 453–471.
- [8] Elmeleegy K., Cox A.L, Ethernet Proxy: scaling Ethernet by suppressing broadcast traffic, *IEEE INFOCOM* (2009).
 - [9] de Olivera R.L.S, Shimoda A.A, Schweitzer C.M, Prete L.R, Using mininet for Emulation and prototyping Software-Defined, *IEEE Colombian Conference on Networks, Communications and Computing* (2014), 1-6.
 - [10] Shpiner A., Keslassy T., Arad C., Mizrahi T., Revah Y., SAL: Scaling Data Centres using smart Address Learning, Technical Report TR 14-02, Technion (2014).
 - [11] Greenberg A., Hamilton J.R, Jain N., Kandula S., Kim, C. Lahiri P., Maltz D.A, Patel P., Sengupta S., VL2: A scalable and flexible data centre network, *communications of the ACM* 54(3) (2011).
 - [12] Mysore R.N, Pamboris A., Farrington N., Huang N., Miri P., Radhakrishnan S., Subramanya V., Vahdat A., Portland: A scalable fault-tolerant layer 2 data centre network fabric, *SIGCOMM '09* (2009).
 - [13] Xiaocui Sun, Zhijun Wang, An effective Caching on Forwarding Table Scheme for Metro Ethernet, *International Journal of Advanced Science and Technology* 23 (2010).
 - [14] Hyunjeong Cho, Saehoon Kang, Younghee Lee, Centralized ARP proxy server over SDN controller to cut down ARP broadcast in large-scale data center networks, *IEEE*.

