An Efficient Address Resolution Technique for Large Layer 2 Networks

ROBERT GILLESPIE, ABDULLAH KAMIL, CHUNG-HORNG LUNG, AND SHIKHARESH MAJUMDAR
Carleton University, Ottawa, Ontario Canada
and
PETER ASHWOOD-SMITH
Huawei Technologies Canada, Ottawa, Ontario, Canada

This paper proposed a Distributed Address Resolution Protocol (DARP) for large layer 2 Ethernet networks used in a data center. Ethernet by design broadcasts Address Resolution Protocol (ARP) messages to all nodes in the same network. As data centers continue to grow in size, there is an increased amount of overhead required to resolve network addresses using the traditional ARP. DARP attempts to reduce this overhead for large data centers with thousands of nodes and allow for the resolution of network address with minimal strain on the underlying network infrastructure. By using Distributed Hash Tables (DHTs) and the existing Chord protocol as the core technologies to maintain address records, we designed a decentralized and reliable service that trades the sporadic overhead associated with current approaches with a consistent and predictable overhead. To determine the viability of the protocol, a series of simulations were developed and run via the OPNET Modeler software package. The simulation results demonstrate that DARP outperforms by a significant margin ARP by reducing the number of messages.

Keywords: Data centers, address resolution protocol, distributed hash tables, distributed hash tables, Chord

1. INTRODUCTION

Address Resolution Protocol (ARP) [Plummer. 1982] is a standard protocol that is used to resolve Internet Protocol (IP) Addresses to Media Access Control (MAC) Addresses in local networks. In order to direct a packet from one machine to another within the same network, the MAC address of the destination machine has to be known by the source machine. The MAC address is the physical address of a machine on the network and each machine has a unique MAC address. An Internet Protocol (IP) address on the other hand is a hierarchical address associated with the addressable entity. In other words, IP addresses are assigned as a part of the operations performed for connecting to a network, and MAC addresses are assigned at the time the hardware is manufactured.

Whenever a machine wishes to communicate with another one in a network, the ARP protocol broadcasts a MAC address request to all machines in the network and whichever machine is using the requested IP address will respond to the requester with an ARP response message containing its MAC Address.

As data centers continue to grow in size, address resolution of IP addresses to MAC addresses using the standard (ARP) is becoming less and less efficient. In addition, in a data center environment, a large number of virtual or physical machines can join or leave the network at any given time. Flooding the network with packets causes an increase in the networks latency, a high bandwidth usage, and possibly congestion in the network. ARP floods the network with a large number of packets even for only one MAC address request, which will generally cause an

Authors' addresses: Authors 1-4, Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario Canada, K1S 5B6; Peter Ashwood-Smith Huawei Technologies Canada, Ottawa, Ontario, Canada K2K 3J1.

increase in network latency and congestion. In large data centers, the number of ARP request packets that are generated in order to get one MAC address are exactly equal to the number of nodes (computers, virtual machines, and servers) in the network which can comprise hundreds of thousands nodes, each hosting multiple virtual machines (VMs). Consider 100,000 servers in a data center, each containing 32 VMs. This can translate to millions of IP and MAC addresses in a data center. In such a large network, the use of ARP creates a very large number of packets for only addresses resolution.

To mitigate the problem of flooding the network with a large number of ARP request messages in a cloud, this paper presents Distributed Address Resolution Protocol (DARP. The core technology that is used for DARP that replaces the ARP is based on Distributed Hash Tables (DHTs)) [Stoica et al. 2003] . Employing DHTs help in reducing the number of ARP request packets in resolving addresses because of the main features of DHTs that are decentralization, fault tolerance, and scalability. DARP can reduce the number of packets for address resolution from N (where N is the number of nodes in a network) to log(N) using DHTs for some use cases. This large reduction in the number of packets solves the issues that ARP is creating such as high bandwidth usage and an increase in network latency and congestion.

The contributions of this paper include: First, we have made some changes to DHTs for the proposed DARP that is used for layer 2 networks in data centers to replace the traditional ARP. Second, we have constructed the DARP protocol, including the message format for various message types that are used for the protocol. Third, we have identified and analyzed a number of use cases for DARP. The control flow of each use case is designed and evaluated. Next, Simulations of the DARP have been performed using OPNET [OPNET]. Compared to the traditional ARP, the results show that DARP can reduce the number of messages sent for address resolution. The reduction can be significant for large layer 2 networks used in large data centers. In [Gillespie et al.], we presented the basic concept of adopting of DHT for DARP and showed some simulation results. This paper elaborates in more detail on the protocol design and demonstrated more experimental results.

The rest of the paper is organized as follows. Section 2 describes the related background about Chord DHTs. Section 3 discusses the DARP protocol design. Section 4 presents the simulation experiments and the results. Section 5 is the conclusion.

2. RELATED BACKGROUND

Data centers are facilities that are used to store, manage, or control data and information of an organization. The size of the data centers usually depends on the size of the organization and the data need to be stored, managed, or controlled. Large data centers can have hundreds or thousands of machines. Each of the physical machines typically runs multiple VMs. Data centers are crucial for cloud computing.

The core technology that is employed by the DARP is the Distributed Hash Table. After investigating a variety of DHT implementations, the Chord [Stoica et al. 2003; Kaffille et al.] protocol was determined to be the best fit for DARP. Chord provides a decentralized, fault-tolerant, and scalable method in which to store information relevant to the resolution of network addresses. Some relevant approaches were examined before we selected Chord. Unstructured peer-to-peer (P2P) approach, e.g., Gnutella [Ripeanu 2001], is based on flooding which may cause loops and result in high level of message overhead and inconsistent search results [Steinmetz et al. 2005]. Pastry [Rowstron et al. 2001] and Tapestry [Zhao et al. 2004] are alternative DHT protocols using structured P2P overlays. But both Pastry and Tapestry provided extra functionality that was either unnecessary or over-complicated for the problem domain. The rest of this section briefly describes key features of Chord DHT protocol, since Chord DHT is the core data structure adopted for our proposal DARP. A detailed description of Chord DHT protocol can be found in [Stoica et al. 2003].

Data Storage The Chord DHT protocol is able to store and distribute data amongst a set of

participating members or nodes. This is performed by using a hash function to map key identifiers to corresponding data. Given some data, the data value is passed to a hashing function. This function then generates a key value which corresponds to the provided data. This resulting (key, value) pair is finally stored within the distributed network of peers to be later retrieved when provided with the correct key value [Stoica et al. 2003].

In order to ensure that the correct data would be recovered when the corresponding key is provided, or that data is not overwritten within the DHT, a consistent hashing algorithm with a sufficiently large key-space must be used. This ensures that, unless malicious intent is involved, it is extremely unlikely that any two pieces of data would generate the same hash value when processed through the hash function. However, if a collision does occur due to the unknown table size or imperfect hash function, collision resolution schemes can be applied to store the keys in distinct buckets [Stoica et al. 2003].

Chord Key-Space - In order to keep track of which members of the Chord ring are responsible for particular (key, value) pairs, the Chord protocol creates an overlay network. This overlay network is structured as a virtual ring topology and member locations are determined by their location in the Chord key-ring. This key-ring is a circular and sequential representation of all possible key values that can be returned by the hash function. Each member of the DHT has an associated position on the key-ring and that position is determined by the member's identifier. This identifier can either be randomly assigned, provided by an administrator, or created by some other means.

Since each participating member of the DHT has an associated position within the key-space, we can define the successor and predecessor nodes within the DHT. The successor to a member is the node whose identifier is next entry within the key-ring. The predecessor member is the node whose identifier is the previous entry within the key-ring. We can also define an ownership of (key, value) pairs that are stored within the DHT. A member owns all pairs whose key value lies between the node's identifier and its predecessor's identifier. Should any node leave the DHT, the members successor would then gain ownership of the leaving node's (key, value) pairs [Stoica et al. 2003].

Data Retrieval - Since all participating members of a DHT are distributed around the virtual overlay network with their position predetermined by the members identifier, the Chord protocol is able to quickly navigate to the member who is storing the desired data given a corresponding key value. To enable this service, the Chord protocol uses the notion of finger tables. A finger table is a list of key-space locations and contains *log(n)* entries where n is the number of possible keys in which the hash function could produce. The i^{th} entry contains the owner of the key value $(id + 2^{i-1} \bmod 2^n)$, where id is the identifier of the member [Stoica et al. 2003]. This feature provides a quick method in which to find a particular location within the key-space, and by extension any value that is stored within the data structure.

Chord for DARP - Efficiency and simplicity were the main reasons that Chord DHT was adopted for DARP. In the typical data center environment where DARP would be deployed, it is likely that a large number of virtual or physical machines would be joining or leaving the network in a short period of time. For this reason, the DARP implementation must be easy to maintain and extremely fault tolerant. The Chord protocol provides us with this flexibility and reliability [Ghodsi ; Hu et al. 2003].

In order for DARP to provide an address resolution service, the protocol must be able to take a given IP address and match it to the corresponding MAC address. Chord provides this service by allowing us to save associated *IP* and *MAC* addresses as (*key*, *value*) pairs. The IP address can be provided to the hash function to provide a consistent key value to reference the MAC address within the hash table. This creates a very simple and effective means of resolving network addresses.

Although extremely unlikely, it is possible to have hash collisions occur while running Chord. This problem becomes non-existent in the DARP setting. By using the entry's IP address as

the hash function input, it is possible to choose a hashing algorithm that will not produce any duplicate hash values, and ensure that collisions within the hash table are not possible. The next section presents the protocol design.

3. DARP DESIGN AND IMPLEMENTATION

This section describes how the idea of DARP is designed and validated for a data center where a large number of machines are connected at layer 2. To design and analyze the protocol, there are some practical issues that DHT needs to be modified to meet the requirements of protocol initialization and communications in a data center environment. DARP was then implemented and validated in the OPNET network simulator. The following sub-sections describe the adaptations that have been made to DHTs for the purpose of cloud network infrastructure.

3.1 DHT Adaptations

A number of adaptations were made to the Chord protocol to simplify it and make the protocol better suited to address resolution in large layer 2 networks. These technical changes include implementing the protocol on layer 2 as opposed to layer 3 or higher, centralizing and managing connection to the DHT ring, and removing the need for a hash table. Each of these technical adaptations is described in the following sub-sections.

3.1.1 *Layer 2 Design and Implementation.* Almost all current implementations of the Chord protocol lie at Layer 3 or higher on the OSI model. In order to create an address resolution protocol for large layer 2 networks, we had to re-design and re-implement the Chord protocol at layer 2. This caused a great number of challenges. The greatest challenges related to this design and implementation were that it was necessary to rewrite algorithms which were designed using remote method invocation, and provide functionality to manage timeouts of unacknowledged messages that was previously handled by lower level protocols.

3.1.2 *Centralizing Bootstrap.* The traditional implementation of Chord allows for any node that wishes to join the DHT ring to request admission from any current member of the ring. In an automated environment, like that of a data center, this is a very difficult task. Any algorithm to create a single unified DHT from neighbouring rings did not provide a fast and reliable method. To avoid this problem all together, we had decided on deploying a centralized bootstrap server into the target network environment. By doing so, DARP is able to create a single unified DHT ring for all members of the network. In addition, it is also possible to deploy multiple bootstrap servers to create multiple virtual local area networks (VLANs) contained within a single large layer 2 network.

The bootstrap server ensures that, unless the node is malicious, no two nodes have the same IP address and be connected to the DHT. The DHT ring is then able to provide the functionality to guide a member to its destination. The bootstrap server provides a bootstrap method of joining into the DHT. It also provides functionality for regulating the distribution of IP addresses across the network, providing DHCP like functionality.

When a node sends a *Join* message (see subsection 3.2) to the server, the server must provide the node with the DHT information, its IP address, and any other networking information (gateway, subnet, broadcast, etc.). If the node is requesting an IP address, the server finds the next available IP address and ensures it is available via a Get function. If the node has a static IP address, the server will check to see if the IP address is already in use and return DHT information. The bootstrap server will return an error if the IP is already in use.

3.1.3 *Managing Concurrent Requests.* One of the major problems that was encountered while working on DARP was managing concurrent join requests made to the bootstrap server. In the initial implementation the bootstrap made no effort to ensure that a member had completed all join operations before permitting another member to begin their join. This had caused the DHT to become fractured where many DHT rings were created around the bootstrap server. This

meant that address resolution and DHT management operations were not able to complete. In the current implementation, the bootstrap server only allows one joining member to go through the sequence of steps required to join the Chord ring. Once that member had completed, the server would allow the next queued member to complete the *join* operations. This implementation is not optimal in terms of efficiency, but it allows for demonstration of the protocol in a stable environment.

3.1.4 *Replacing DHCP.* When creating DARP, another practically important issue is that the Dynamic Host Configuration Protocol (DHCP) relies on ARP. DHCP is a standard communications protocol that allows network administrators to manage the assignment of IP address in an administrative network.

Without ARP, DHCP is unable to determine if there is a host on the local network with a statically assigned IP address. DARP therefore provides functionality to replace DHCP and dynamically allocate IP addresses to other nodes in the network. This operation is performed during member connection to the DHT ring. When a member requests to join the DHT, it has the option to request that the bootstrap server assign the member an IP address dynamically or it will provide the bootstrap server the desired static IP address. If the member provides an IP address, the bootstrap server will check the DHT ring to see if that IP address has already been allocated. If the address is being used by another node within the network, the node will not be permitted to join the overlay network. This added functionality allows for the removal of DHCP in favor of a more configurable alternative.

3.1.5 *Removing the Hash Function.* One of the more interesting changes that was made to the Chord protocol was that we found a way to remove the necessity for a hash function. In DARP, we have substituted the hash functions key-space for the IP address-space. By doing so, the protocol is able to ensure that there are no collisions in the data structure, as IP addresses are unique, and hence that all entries would be evenly distributed around the DHT ring. If a hash function is added, there will be added overhead in processing and the resulting hashes will at least four bytes in length or longer.

Using the new design, a members IP address would become its key-space identifier and it would store its own IP/MAC address (key, value) pair. Since there is exactly the same number of participating nodes in the overlay network as there are number of entries to be stored in the DHT, we can ensure that each member of the DHT stores exactly one record, its own IP/MAC address pair. Hence, the Chord key space used for this protocol is 32-bits in length and corresponds to the IP Address of the Node it represents. Figure 1 illustrates this idea.

3.2 Packet Format

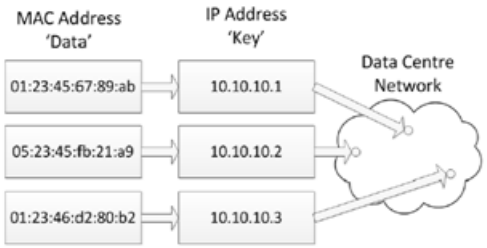


Figure. 1: Storing IP/MAC address pairs in DARP.

To facilitate communications between the different instances of DARP servers within the simulations, it is necessary to create a set of packet formats for various message types. The messages represent functions in the Chord framework. Whenever a message is sent, an ACK is expected to be returned within 500 ms. Each message is contained within an Ethernet Frame.

In the OPNET software, these packet formats define the fields that exist within the messages. This includes their names, types, and lengths (in bits). For DARP, the set of message types that are part of the protocol includes: *Join*, *Join-Response*, *Reconnect*, *Leave*, *Get MAC*, *Reconnect*, *Successor Request*, *Predecessor Request*, *Set Successor Request*, *Set Predecessor Request*, *Update MAC*, *Heartbeat*, *Connected*, and *ACK*. Table I shows the packet format used for the header of messages being sent by DARP. This header includes the MAC addresses of the source and destination, the IP address of the source, the op code identifying the packet type, and the acknowledgment number for the message. This format is used for all messages that do not require additional content.

Table I: The basic DARP packet format

Source MAC address (48 bits)	
Destination MAC address (48 bits)	
Source IP address (32 bits)	op code (8 bits)
ACK number (32 bits)	

In the cases where additional information is required in the packet, a set of specialized packet formats were created. In total, there are eight of these formats. Table II contains an example of the expanded packet formats. This figure corresponds to the *Response to Join* packet format. In addition to the common header used in all DARP packets, this packet contains all of the necessary network interface information required by the node in order to communicate on the network. Detailed descriptions of packet formats, including the payload for various message types, can be found in [**?**].

Table II: The DARP *Response to Join* packet format

Source MAC address (48 bits)	
Destination MAC address (48 bits)	
Source IP address (32 bits)	op code (8 bits)
ACK number (32 bits)	

Successor MAC address (48 bits)
Node IP address (32bits)
Gateway IP address (32 bits)
Network IP address (32 bits)
Subnet IP address (32 bits)

3.3 Use Cases

A number of use cases have been considered and analyzed to help the design of DARP. Identification of use cases is an important and effective mechanism for system evaluation. We follow the UML standards [**?**] for use case identification and specifications. The key use cases include *Join DHT*, *Leave DHT*, *Get MAC Address*, *Update MAC Address*, *Register Node*, *Set Predecessor*, and *Set Successor*. The use cases are closely related to the message types designed for communication protocol designed in this paper. Use cases can specify both normal sequence of flow and exceptional cases. Three critical and representative use cases in normal conditions are presented in this paper. Table III demonstrates the sequence of flow for the normal *Join DHT* use case, Table IV shows the use case of normal *Get MAC Address*, and Table V illustrates the normal *Set Predecessor* use case. Detailed descriptions of abnormal alternative flows of these two use cases and other use cases are documented in [Gillespie et al. 2012]. In the use cases, 0.4 seconds if used, are configurable values for system timeout.

Table III: Normal *Join DHT* use case and sequence flow

Brief Description	The Node requests access to the DHT from the Bootstrap Server.
Precondition	The Node is running. The Bootstrap Server is running.
Primary Actor	Node
Secondary Actors	None
Dependency	INCLUDE USE CASES Register Node, and Establish DHT Connections & Variables
Generalization	None

Basic Flow	
Step	**Action**
1	*The Node requests that the system adds the Node to the DHT.*
2	*The system sends a Join message to the Bootstrap Server.*
3	*INCLUDE USE CASE Register Node.*
4	*The system VALIDATES THAT a response is received within 0.4 seconds.*
5	*The system VALIDATES THAT the response is a Join-Response message.*
6	*The Node instantiates its local variables.*
7	*The Node requests an initial Finger Table.*
8	*The system sends a Finger Table Request to the Node's successor.*
9	*INCLUDE USE CASE Send Finger Table.*
10	*The system VALIDATES THAT a response is received within 0.4 seconds.*
11	*The system VALIDATES THAT the response is a Finger Table ACK message.*
12	*The Node fills in its Finger Table.*
13	*The Node requests the MAC Address of its predecessor from the system.*
14	*The system sends a Predecessor Request to the Node's successor.*
15	*INCLUDE USE CASE Send Predecessor*
16	*The system VALIDATES THAT an ACK message is received within 0.4 secs.*
17	*The system VALIDATES THAT the ACK message is of type Success-Predecessor.*
18	*The Node sets its predecessor.*
19	*The system sends a Set Predecessor Request message to the Node's successor.*
20	*INCLUDE USE CASE Set Predecessor.*
21	*The system VALIDATES THAT an ACK message is received within 0.4 secs.*
22	*The system VALIDATES THAT the ACK message is of type Success.*
23	*The system sends a Set Successor Request message to the Node's predecessor.*
24	*INCLUDE USE CASE Set Successor.*
25	*The system VALIDATES THAT an ACK message is received within 0.4 secs.*
26	*The system VALIDATES THAT the ACK message is of type Success.*
Postcondition	*The Node is integrated into the DHT.*

4. SIMULATIONS AND RESULTS

This section discusses simulations for Address Resolution Protocol (ARP) and Distributed Address Resolution Protocol (DARP). Simulations have been performed to (i) validate the concept of DARP and (ii) compare ARP and DARP on large networks and different types of networks. OPNET was used to perform the testing and comparison. The following first describe network topologies that are used our simulations, which is followed by simulation results.

4.1 Data Centers Network Topologies

Although data centers can have many topologies, which depend on the size of the network, only fat tree and mesh topologies are mainly used in this paper. These two topologies are adopted, because they are the most common ones in large data centers.

Fat Tree Topology [Al-Fares et al. 2008; Mysore et al. 2009] is a common topology used in large data center networks. This topology uses a routing algorithm that balances the network traffic as well as reduces the delay time. In fat tree networks, two-way links are usually used with the same

Table IV: Normal *Get MAC* Address use case and sequence flow

Brief Description	*The Node queries the DHT for the MAC Address of a given IP Address.*
Precondition	*The Node is a part of the DHT.*
Primary Actor	*Node*
Secondary Actors	*None*
Dependency	*INCLUDE USE CASE Get MAC Address*
Generalization	*None*

Basic Flow	
Step	**Action**
1	The system determines the closest preceding Node to the quested MAC Address from the Finger Table.
2	The system VALIDATES THAT the Node is NOT the closest preceding Node in the DHT.
3	The system sends a Get message to the closest preceding Node.
4	INCLUDE USE CASE Get MAC Address.
5	The system VALIDATES THAT an ACK message is received within 0.4 secs.
6	The system VALIDATES THAT the ACK message is of type 'Success -Lookup Result'
7	The system adds the IP Address/MAC Address key/value pair to the Node's address cache.
8	The system provides the Node the MAC Address for the requested IP Address.
Postcondition	The Node receives the requested MAC Address.

Table V: Normal *Set Predecessor* use case and sequence flow

Brief Description	*The Node sets its predecessor to the provided address.*
Precondition	*The Node received a Set Predecessor Request message.*
Primary Actor	*Bootstrap Server*
Secondary Actors	*Node*
Dependency	*None*
Generalization	*None*

Basic Flow	
Step	**Action**
1	The Node stores the MAC Address of its new predecessor.
2	The system sends an ACK message indicating success to the Node that sent the message.
Postcondition	The Node's predecessor's MAC Address is updated.

capacity in the binary tree and the number of links (or bandwidth) is reduced to half as you go up to the root (see Figure 2). The main characteristic of the fat tree topology is fixed bisectional bandwidth. Fat tree is very efficient in terms of time delay and traffic balancing, which makes it attractive for large data center networks [14, 15].

Mesh Topology is also a topology type that is used in large networks or data centers. In this topology, each node does not only receive packets, but is also used as a relay for other nodes in a network. Each node has more than one connection, which makes the network more reliable. In other words, if one connection is down, there is another path for each source and destination. Although the topology is more reliable than other topologies, it is more expensive to operate or maintain. Mesh is mainly used in this paper to compare the performance of ARP and DARP as a second topology.

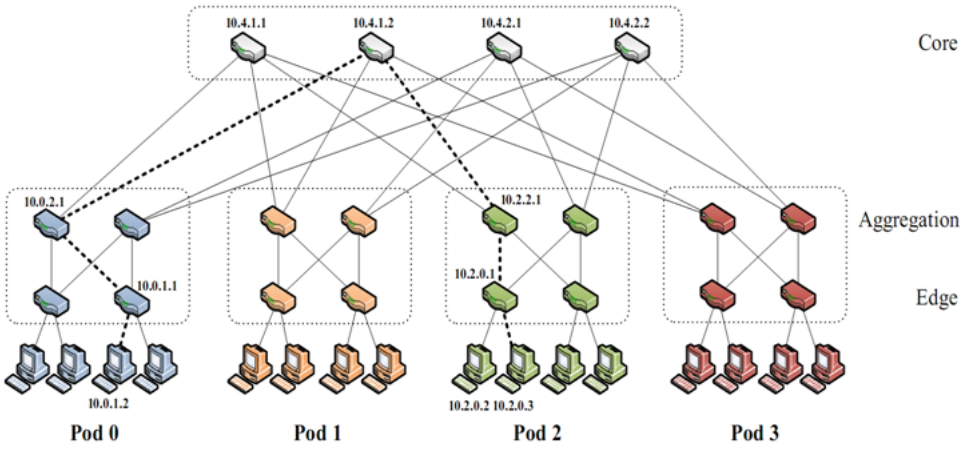


Figure. 2: A common fat tree network [14, 15].

4.2 OPNET Simulations

The traffic profile used in the simulations was the default traffic in OPNET [3], which includes Database (Heavy), File Transfer (Heavy), and Web Browsing (Heavy). All servers in the networks used this traffic profile as their background traffic. All servers also act as database and FTP servers. All three types of traffic (Database, File Transfer, and Web Browsing) were used for each simulation to make the network congested at some point to compare the results.

Several topologies have been built for testing in OPNET. Three of those networks (Networks I, II, and III) are presented in this paper. Figure 3 shows network I, a simple fact tree network for illustration. There are 8 servers at the bottom layer. The simulation was conducted for several hours (simulated time) of traffic flows between all nodes. Both scenarios for ARP and DAPR have exactly the same input traffic. Figure 4 illustrates Network II. The network consists of 108 servers, 9 switches, and one router (the root). The simulation was for one hour (simulated time) of traffic flows between all nodes. Both scenarios for ARP and DAPR have exactly the same traffic. Figure 5 depicts Network III, one of the mesh networks used in the simulation. The network consists of 128 servers. The simulation duration for this network is one hour (simulated time). Again, both scenarios for ARP and DAPR have identical traffic.

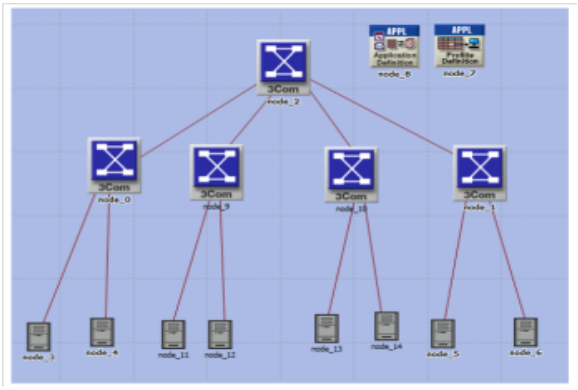


Figure. 3: Network I: a simple fat tree network tested

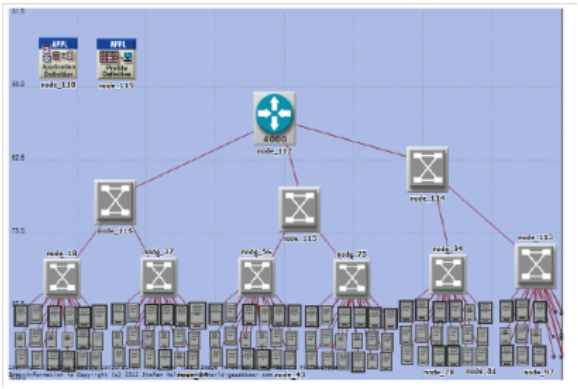


Figure. 4: Network II: a fat tree network tested

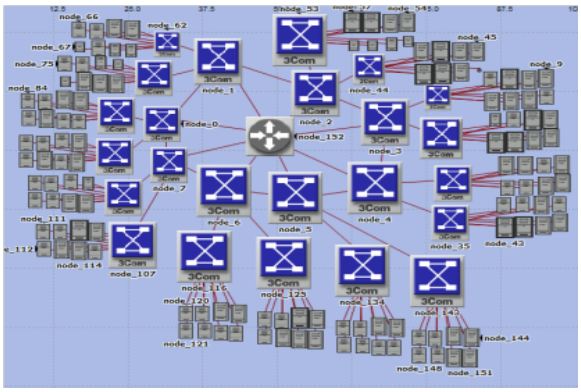


Figure. 5: Network III: a mesh network tested

4.3 Simulation Results

Simulations using OPNET were performed to analyze the behaviour and performance of the proposed protocol. A representative set of simulation results obtained from different network topologies is presented here to compare the performance of ARP and DARP.

Figure 6 presents the delay in the network for Network I. The x-axis labels the simulation dates/times and the y-axis is the Ethernet delay (in seconds). Although the network is small and ARP works efficiently (upper blue curve) with small networks, DARP demonstrates an even higher performance (lower red curve) and is more predictable compared to the traditional ARP. The delay observed with DARP seems to be insensitive to the change in time and it outperforms ARP by as much as 42%.

Figure 7 shows the number of packets (y-axis) per second flowing between two nodes, and the x-axis is the simulation duration (in simulated hours). It is evident that the number of ARP packets is significantly higher than that of DARP, i.e., from 700-850 packets per second down to 300 packets per second. Further, the performance of DARP seems to be fairly insensitive to the simulated time.

Figure 8 shows the average number of ARP packets sent per second at a node, node 0, for Network II as shown in Figure 4. The lower red curve represents DARP and the upper blue curve represents ARP. The average number of packets for address resolution using the DARP protocol ranges from 3,000 to 5,000 packets per second while using the ARP protocol generates more packets which range from 7,000 to 8,000 packets per second. In other words, the traditional ARP protocol sends more packets for address resolution than the proposed DARP. Note that the x-axis is the simulation duration (in minutes) and the y-axis is the load at node 0 (in packets per

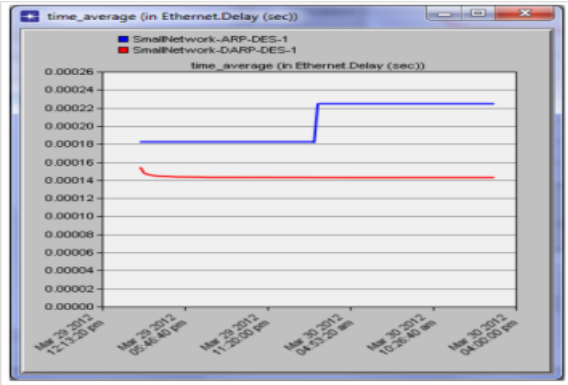


Figure. 6: Ethernet delay (sec) for Network I

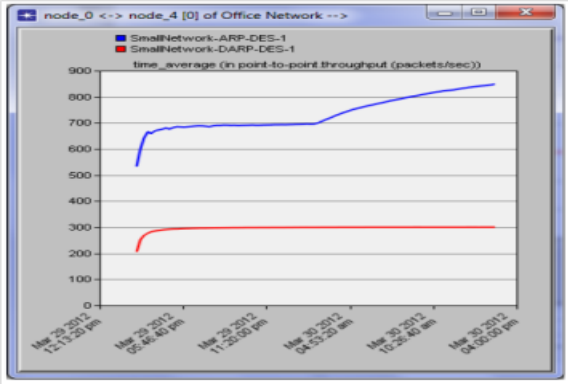


Figure. 7: Number of packets/sec sent between two nodes for Network I

second).

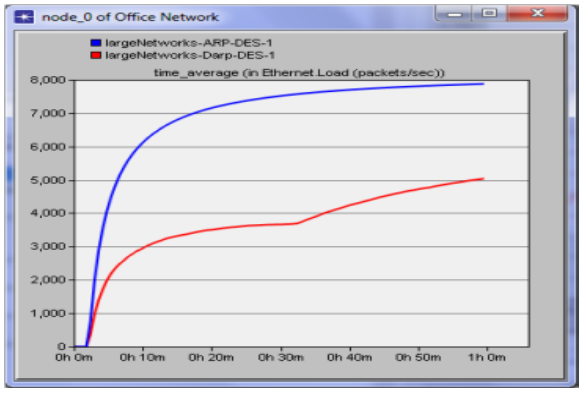


Figure. 8: Node load (packets/sec) for Network II

Figure 9 shows the throughput (as shown in y-axis in packets per second) in the link between node 1 and node 18 as an example. The lower red curve represents DARP and the upper blue curve represents ARP. It can be seen that the traffic between any nodes with DARP is much less than the ones with ARP. This means that ARP generates more traffic due to address resolution. The x-axis is the simulation duration (in minutes).

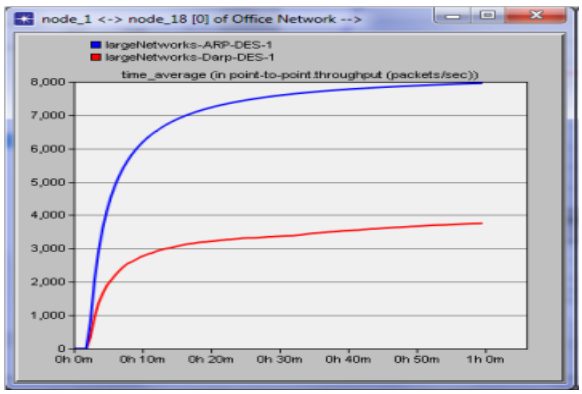


Figure. 9: Throughput between two nodes (packets/sec) for Network II

Figure 10 shows the utilizations for the link between node 1 and 18 for illustration. The lower red curve represents the DARP and the upper blue curve represents ARP. The average utilization (in percentage in y-axis and x-axis shows the simulated time elapsed in munites) for the links between node 1 and node 18 using the DARP protocol is around 40%, but with the ARP protocol the average utilization reaches almost 95%. The results demonstrate that DARP is more efficient than ARP in large layer 2 networks. More fat tree networks with different sizes were constructed and tested. All results show that DARP outperforms ARP.

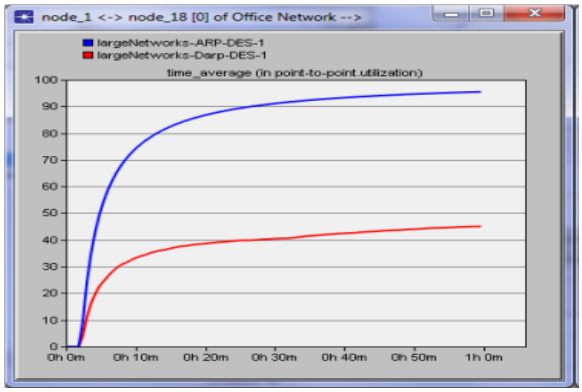


Figure. 10: Link utilization (%) for Network II between two nodes

The results in Figures 11 and 12 also demonstrate that DARP works better than ARP for Network III, a mesh topology, as diagrammed in Figure 5. The lower red curve represents the DARP and the upper blue curve represents ARP In Figure 11, the x-axis represents the simulated time elapsed in minutes and the y-axis is the link throughput (in packets per second). In Figure 12, the x-axis represents the same and the y-axis is the link utilization (in percentage). As can be seen in both figures, the performance of DARP is better than that of ARP; the largest gap between these two protocols is about 2,000 packets per second or 20% for link utilization.

Both figures also show that the number of messages decreases for ARP after about 30 minutes. This is mainly due to the fact that the node degree (the average number of connections of a node) in a mesh topology is high and more MAC request messages are broadcast over the network due to the mesh connection. The MAC addresses are returned to the sender, which resulted in more MAC addresses have been cached locally; therefore, some Get MAC messages do not have to been sent over the network.

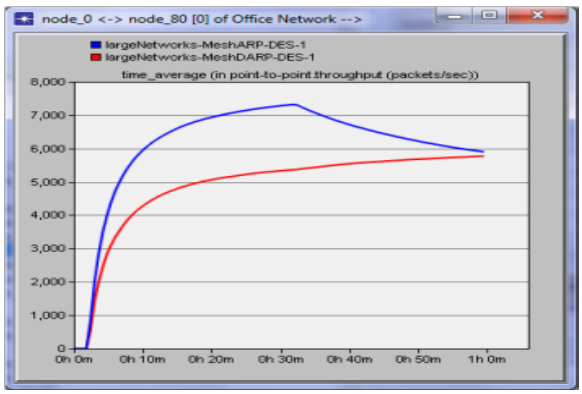


Figure. 11: Link throughput (packets/sec) for mesh Network III

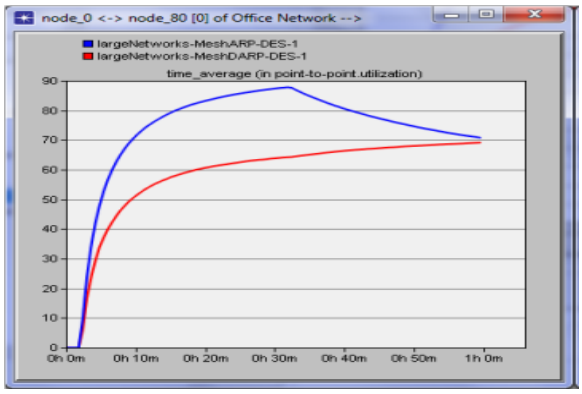


Figure. 12: Link utilization (%) for Network III

5. CONCLUSIONS AND FUTURE RESEARCH

This paper presented a new protocol, DARP, for address resolution for large layer 2 networks that are used in data centers. As data centers grow, ARP is becoming less efficient, because its broadcast mechanism increases the number of messages for the network and increases delay in packet delivery. This could cause serious performance issues in data centers that comprise a large number of physical servers each of which running of multiple virtual machines. The new protocol was designed using the DHT technology and several features for ARP, and DHTs have been specifically tailored for layer 2 networks used in a data center. Employing the DHT technology resulted in a reduction of the number of APR request packets from N to $log(N)$, where N is the number of nodes in a network. A number of simulation experiments have been conducted with OPNET. The simulation results demonstrate that DARP is capable to work in both small and large networks with a smaller delay and a lower number of messages compared to those obtained from ARP. The performance of DAPR is also insensitive to different topologies. Based on simulation results reported in Section 4.3, a decrease of about 42% in delay is observed when ARP is replaced by DARP for Network I. A significant decrease in the number of messages accrues from the replacement of ARP by DARP for all networks experimented. As discussed in Section 3.3, DARP also handles a variety of different uses cases. These include *Join DHT*, *Leave DHT*, *Get MAC Address*, *Update MAC Address*, *Register Node*, *Set Predecessor*, and *Set Successor*.

In our future work on DARP, considerations should be made to enable multiple members to join the DHT at the same time. Practically, this scenario can occur for a large number of machines in a data center. Another direction to investigate is the impact of a large number of dynamic

join and leave operations for a data center environment.

REFERENCES

Plummer, D. 1982. An Ethernet Address Resolution Protocol, *IETF RFC 826*, Nov. 1982.

Stoica, I. Morris, D. Liben-Nowell, D. Karger, D.R. Kaashoek, M.F. Dabek, F. and Balakrishnan, H. 2003. Chord: a scalable peer-to-peer lookup protocol for internet applications.. *IEEE ACM Trans* on Networking, 11(1), 17-32.

OPNET. `www.opnet.com`.

Gillespie, R., Kamil, A., Lung, C.H., Majumdar, S. and Ashwood-Smith, P. 2005. Address resolution in large layer 2 networks for data centers. In *Proceedings of the 4th IEEE International Confrence on Cloud Computing Technology and Science*. (CloudCom). 693-698.

Kaffille, S., and Loesing, K. 2007. Open Chord version 1.04 Userś Manual. `http://open-chord.sourceforge.net`.

Ripeanu, M. 2001. *Peer-to-peer architecture case study: Gnutlella*. In *Proceedings of International Confrence on Peer-to-Peer Computing*. 99-100.

Steinmetz, R., and Wehrle, K. 2005. Peer-to-Peer Systems and Applications. Springer-Verlag Berlin Heidelberg.

Rowstron, A., and Druschel, P. 2001. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of International Confrence on Distributed Systems Platforms*.329-350.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. D. 2004. Tapestry: A resilient global-scale overlay for service deployment. *IEEE* Journal on Selected Areas In Communications, 22(1). 41-53.

Ghodsi, A.. Distributed k-ary System: Algorithms for Distributed Hash Tables,. *Doctoral thesis* KTH-Royal Institute of Technology.

Xu, Z., Min, R., and Hu, Y. 2003. Reducing maintenance overhead in DHT based peer-to-peer algorithms. In *Proceedings of Third International Conference on P2P Computing*. 218-219.

Gillespie, R., and Kamil, A. 2012. Address Resolution in Large Layer 2 Networks. *4th Year Project Report* Dept. of Systems and Computer Engg., Carleton Univ., April 2012.

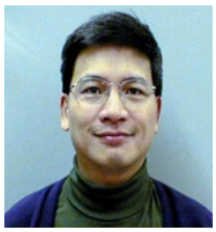
UML. Unified Modeling Language,. `http://www.uml.org/`.

Al-Fares, M., Loukissas, A., and Vahdat, A. 2008. A scalable, commodity data center network architecture. In *Proceedings of ACM SIGCOMM Confrence* on Data Communication. 63-74.

Mysore, R.N., Pamboris, A., Farrington, N., Huang, P., Radhakrishnan, S., Subramanya, V., and Vahdat, A. 2009. PorLand: A scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of SIGCOM 2009 Conference on Data Communication. 39-50*.

Robert Gillespie received his B.Engg. degree in Software Engineering in Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada. He is a Software Developer at Kinaxis, Ottawa. He is interested in Software Engineering, Computer Systems, and Computer Networks.

Abdullah Kamil received a B.Eng. degree in Communications Engineering from Carleton University, Ottawa, Canada. He is currently a software verification engineer at Ciena, Ottawa Canada. He has been working on software verification for carrier Ethernet switches. His professional interests focus on cloud computing and telecommunication technologies.



Chung-Horng Lung received the B.S. degree in Computer Science and Engineering from Chung-Yuan Christian University, Taiwan and the M.S. and Ph.D. degrees in Computer Science and Engineering from Arizona State University. He was with Nortel Networks from 1995 to 2001. In September 2001, he joined the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, where he is now an associate professor. His research interests include Software Engineering and Communication Networks.

Shikharesh Majumdar is a Professor and the Director of the Real Time and Distributed Systems Research Centre at the Department of Systems and Computer Engineering at Carleton University in Ottawa, Canada. He is also associated with the *Huawei-TELUS Centre of Innovation for Enterprise Cloud Services* that focuses on collaborative research among Huawei, TELUS and Carleton University. Dr. Majumdar is a member of the faculty team actively involved with Carleton University's Canada-India Centre for Excellence. He holds a M.Sc. and a Ph.D. degree in Computational Science from University of Saskatchewan, Saskatoon, Canada. Before his graduate studies in Canada he did a Bachelor of Electronics and Telecommunications Engineering and a Post-Graduate Diploma in Computer Science (hardware) from Jadavpur University in India and completed the Corso Di Perfezionamento (Master's in Engineering) in Electro-techniques (Communication) from Politecnico Di Torino in Italy. Dr. Majumdar has worked at the R&D Wing of Indian Telephone Industries (Bangalore) for six years. His research interests are in the areas of cloud and grid computing, operating systems, middleware and performance evaluation. Dr. Majumdar actively collaborates with the industrial sector. He is the area editor for the Simulation Modelling Practice and Theory journal published by Elsevier. Dr. Majumdar is a member of ACM and IEEE and was a Distinguished Visitor for the IEEE Computer Society from 1998 to 2001.

Peter Ashwood-Smith has a B.Sc. Computer Science and M.Sc. Computer Science from the University of Toronto, he has worked on the patenting, design, standardization, implementation, deployment, and support of many modern routing protocols including early proprietary label switched networks then standardized protocols like ATM, MPLS, optical switching with GMPLS, and large flat L2 networks with Shortest Path Bridging. His work is deployed in networks all over the world including Data Centers, Enterprises and core networks. Peter has approximately 50 networking related patents, previously he was a Fellow at Nortel Networks and is now employed as a senior researcher with Huawei Technologies Canada.