A NETWORK METADATA INFRASTRUCTURE FOR LOCATING NETWORK DEVICES

A thesis presented to

the faculty of

the College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

David Alexander

March 2004
This thesis entitled

A NETWORK METADATA INFRASTRUCTURE FOR LOCATING NETWORK DEVICES

BY

DAVID ALEXANDER

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Shawn D. Ostermann
Associate Professor of Electrical Engineering and Computer Science

R. Dennis Irwin
Dean, Russ College of Engineering and Technology
ALEXANDER, DAVID. M.S. March 2004  
Electrical Engineering and Computer Science  
A Network Metadata Infrastructure for Locating Network Devices (58pp.)  

Director of Thesis: Shawn D. Ostermann  

Finding the physical location of a device given only its network address is a difficult network management problem which requires collecting and correlating large amounts of data from routers, switches, and other data sources. Routers store a mapping of IP addresses to hardware addresses in an ARP table. Switches keep a mapping between MAC addresses and ports in a CAM table. This work studies the feasibility of collecting, processing, and archiving the contents of these tables. A graphical user interface provides the ability to search through the summarized ARP and CAM data to find the physical location of a device at a given point in time.  

Ohio University’s network, consisting of 15 routers and 600 switches, served as a testbed for the system. Router data collection and processing occurred once per hour and took six minutes to complete. Switch data collection and processing occurred four times per hour and took five minutes per sample. ARP and CAM data for several years was stored in a database using significantly less space than the raw data. Searches for an IP or MAC address made through the graphical interface took fourteen seconds. Historical views of ARP data are useful for IP address space management, while recent ARP data can be used for network security. Intrusion detection systems can identify an IP address involved in a network attack, and the graphical interface can trace the IP address to the switch port, building, room, and user of the device.  

Shawn D. Ostermann  
Associate Professor of Electrical Engineering and Computer Science
ACKNOWLEDGEMENTS

I would like to thank Dr. Ostermann and the Ohio University Communication Network Services staff. Without their help, this work would not have been possible. I would also like to thank my parents and Laura for their patience and support.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>3</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>7</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>8</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>9</td>
</tr>
<tr>
<td>2 SYSTEM DESIGN</td>
<td>13</td>
</tr>
<tr>
<td>2.1 ARP System Design</td>
<td>14</td>
</tr>
<tr>
<td>2.1.1 ARP Data Collection</td>
<td>15</td>
</tr>
<tr>
<td>2.1.2 ARP Database Tables</td>
<td>16</td>
</tr>
<tr>
<td>2.1.3 ARP Processing Algorithms</td>
<td>21</td>
</tr>
<tr>
<td>2.1.4 User Interface</td>
<td>23</td>
</tr>
<tr>
<td>2.2 CAM System Design</td>
<td>24</td>
</tr>
<tr>
<td>2.2.1 CAM Data Collection</td>
<td>25</td>
</tr>
<tr>
<td>2.2.2 CAM Database Tables</td>
<td>26</td>
</tr>
<tr>
<td>2.2.3 CAM Collection and Processing Algorithms</td>
<td>28</td>
</tr>
<tr>
<td>2.3 ARP and CAM Integration</td>
<td>30</td>
</tr>
<tr>
<td>2.4 Test Environment</td>
<td>31</td>
</tr>
<tr>
<td>2.4.1 Network Topology</td>
<td>31</td>
</tr>
<tr>
<td>2.4.2 Database Infrastructure</td>
<td>33</td>
</tr>
<tr>
<td>2.4.3 Clustered Computing Environment</td>
<td>34</td>
</tr>
<tr>
<td>2.4.4 Storage Area Network</td>
<td>34</td>
</tr>
<tr>
<td>3 RESULTS</td>
<td>35</td>
</tr>
<tr>
<td>3.1 Experimental Results and Analysis</td>
<td>35</td>
</tr>
</tbody>
</table>
3.1.1 Router Data Collection and Processing .......................... 35
3.1.2 Switch Data Collection and Processing ......................... 41
3.1.3 Data Presentation and Query Performance ..................... 44
4 CONCLUSIONS ................................................................. 46
  4.1 Related Work .............................................................. 46
    4.1.1 NetReg/NetMon .................................................... 47
    4.1.2 Spectrum ........................................................... 48
    4.1.3 OpenView ......................................................... 49
  4.2 Summary ................................................................. 50
  4.3 Future Work ............................................................. 52
BIBLIOGRAPHY ................................................................. 57
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Arplog Table</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Xref_Interface Table</td>
<td>18</td>
</tr>
<tr>
<td>2.3 Xref_Vlan Table</td>
<td>19</td>
</tr>
<tr>
<td>2.4 Xref_Router Table</td>
<td>19</td>
</tr>
<tr>
<td>2.5 Xref_Manufacturer Table</td>
<td>20</td>
</tr>
<tr>
<td>2.6 Xref_Manufacturer Sample Data</td>
<td>20</td>
</tr>
<tr>
<td>2.7 Arplog_Updates_History Table</td>
<td>21</td>
</tr>
<tr>
<td>2.8 Arplog_Events_History Table</td>
<td>21</td>
</tr>
<tr>
<td>2.9 Database Table Constraints</td>
<td>22</td>
</tr>
<tr>
<td>2.10 Switch Table</td>
<td>26</td>
</tr>
<tr>
<td>2.11 Cam Table</td>
<td>27</td>
</tr>
<tr>
<td>2.12 Cam_Updates_History Table</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Results of Caching for CAM Routers</td>
<td>38</td>
</tr>
<tr>
<td>3.2 Results of Caching for All Routers</td>
<td>39</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Example Trace of an IP Address to a Location</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>Generalized System Architecture</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>ARP Processing Algorithm</td>
<td>23</td>
</tr>
<tr>
<td>2.3</td>
<td>Screen Capture of ARP User Interface</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>Switch Data Collection Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>2.5</td>
<td>Switch Data Processing Algorithm</td>
<td>29</td>
</tr>
<tr>
<td>2.6</td>
<td>Screen Capture Showing Location Information</td>
<td>31</td>
</tr>
<tr>
<td>2.7</td>
<td>Ohio University Network Topology Diagram</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Router CPU Utilization</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>New Arplog Table Entries By Month</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td>Average Query Time By Month</td>
<td>45</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

Finding the physical location and possible users of a network device given only a network address is a difficult network management problem which requires collecting and correlating large amounts of data from routers, switches, and other data sources. This work studies the feasibility of building such a system and analyzes the results of implementing the system on Ohio University’s network.

Computer networks are highly dynamic systems. A network device plugged into a jack can obtain an IP address using the Dynamic Host Configuration Protocol [13] and begin communicating on the local network and global Internet. The DHCP protocol simplifies host configuration, but it complicates network management by making it difficult to determine the location and users of a device at any given point in time.

IP trace-back techniques proposed in [5], [7], [20], and [21], have not been standardized or made available for different platforms. Moreover, many of those proposed techniques are only approximate and require large flows with a significant number of packets to locate a host. These techniques rely on intrusion detection systems to detect an attack within a few minutes and discard the information after a short period of time.

MAC address registration is another approach for tracking the location and users of a network device. CMU’s NetReg/NetMon [9] project implements this approach. A user must register a new device in order to gain access to the network. Once a user registers a device, data collected from routers and switches is used to track the device’s physical location. MAC address registration simplifies network management, making it easier to track users, but it places restrictions on the network and its users.
It is not clear if MAC address registration can scale to a large network with many users.

Ohio University’s network does not require MAC address registration, so other methods must be used to locate devices on the network. Routers and switches store information that can help locate devices. Figure 1.1 shows how we can trace an IP address to a building and room. Given an IP address, we can search through a router’s ARP table to find the device’s MAC address. We can then look at a switch’s CAM table to find the port number where the device is attached to the network. Network inventory information maps a switch port to a building and room number. Possible users of the device can be identified by searching through the university directory by building and room.

We can locate a device in real-time by examining the contents of the router’s ARP table and switch’s CAM table, but the dynamic nature of these tables prevents us from finding the location of a device during some time interval in the past. Router ARP entries persist for four hours, while switch CAM entries remain in the table for five minutes. If we want to answer questions about the past location of a device, we need to collect and correlate ARP and CAM data. There are several challenges to collecting and correlating ARP and CAM data, including finding appropriate collection methods, polling intervals, and aggregation methods. Heterogeneous network architectures consisting of different router and switch models require flexible data collection methods. The choice of a polling interval affects the accuracy of the system, but choosing a polling interval that is too small may negatively impact the routing or switching performance and may generate a data set that cannot be processed before the next sample is collected. Because the ARP and CAM tables can be large, it is prohibitive to store the raw data. The raw data can be aggregated and stored in a database to facilitate retrieval and analysis.

We developed a graphical user interface to display real-time and historical views of the devices on the network. The interface combines ARP, CAM, inventory, and
Figure 1.1. Example Trace of an IP Address to a Location
directory data to locate a device and identify its possible users. Historical data can be used to find network usage patterns and to plan IP address allocation. Recent ARP and CAM information can be integrated with intrusion detection systems to locate devices that are the source or target of a network attack. Users of the compromised device can be contacted, or the switch port can be disabled to remove the affected device from the network.

The ARP and CAM system was implemented and tested on Ohio University’s network. The network consists of 15 routers and 600 switches. Section 2.4 describes the network architecture.

This work is part of a larger effort to build a Network Metadata Infrastructure (NMI) for maintaining aggregated network data collected from various sources. The system for finding the physical location of network devices is a specific instance of a NMI module. The architecture and design process can be generalized as a system for aggregating network metadata.

Chapter 2 describes the system design. Chapter 3 describes the testbed and analyzes experimental results. Chapter 4 draws general conclusions and suggests future improvements.
2. SYSTEM DESIGN

This chapter describes the design of a system for finding the physical location and possible users of a network device using data collected from routers and switches. We built the system in three phases. In the first phase, we developed a system to collect, process, and store ARP data collected from routers. We analyzed the ARP system to determine the feasibility of data collection and the growth rate of aggregated data. In the second phase, we developed a similar system to collect, process, and store CAM table data collected from switches. The final phase integrated these two systems with other data sources so that devices and their users can be located given an IP or MAC address.

Figure 2.1 shows the generalized system architecture. Each system is divided into modules that perform data acquisition, processing, storage, and presentation. This chapter describes the design of each module. A detailed analysis of the system performance is given in Chapter 3. Chapter 4 proposes an improved design based on analysis of the experimental results.

![Figure 2.1. Generalized System Architecture](image-url)
2.1 ARP System Design

The router’s ARP table contains information about active network devices. The Address Resolution Protocol [19] maps a network (IP) address to a data link layer hardware address. When a router needs to resolve an IP address to a hardware address, it creates an entry in its ARP cache with the given IP address and an incomplete MAC address. The router broadcasts an ARP request packet containing the IP address of a destination host. The host with the specified destination IP address sends an ARP reply with its MAC address. If the router receives the ARP reply, it uses the MAC address to send the original packet. In addition to the IP/MAC address mapping, the router’s ARP table contains the router’s IP address, timestamp, and the name of the interface where the IP/MAC address mapping was seen.

The format of a router’s ARP table entry varies by the router’s function. Routers that perform bridging or switching between VLANs have a slightly different ARP cache format than that of a regular router. The ARP cache for a switch-router contains the VLAN number instead of the interface name. The ARP cache for a bridge-router contains the Bridge Virtual Interface (BVI) number instead of the interface name. The switch-router’s CAM table and a bridge-router’s bridge table both map a MAC address to a port number.

In the initial implementation, we combined the switch and bridge router ARP entries with CAM or bridge table entries to give the port where an IP/MAC address binding was seen. The problem with this approach is that the CAM and bridge table timeouts are much shorter than the ARP table timeout.

A router’s configuration may give information about the location of devices serviced by an interface, but the configuration may not contain enough detail to find the exact location of a device. Comments can be added to the router configuration to note a location serviced by an interface. For example, a comment in the configuration maps an interface to a building. A switch attached to the router interface can service rooms within a building. Finding the exact physical location of a network device
requires collecting data from switches on the network and keeping track of network inventory information mapping a switch and port number to a room.

### 2.1.1 ARP Data Collection

We can collect the contents of a router’s ARP cache using SNMP or Expect. SNMP is an application layer protocol used to manage network devices. Expect is a scripting language used to automate interactive applications such as telnet. In this system, we used Expect for ARP data collection, but both SNMP and Expect have similar security and reliability issues.

Authentication, privacy, and access control are all needed to address the security threats faced by SNMPv1, which include Modification of Information, Masquerade, Disclosure, and Message Stream Modification attacks [6]. SNMPv1 lacks privacy and is vulnerable to these threats. SNMPv3, which is described in [15], [10], [16], [6], and [22], provides authentication, privacy, and access control, which can protect against the security vulnerabilities found in SNMPv1. Unfortunately, the campus routers do not support SNMPv3. In the absence of SNMPv3 support, other security considerations should be made.

Since the read-only community string is sent unencrypted on the network, an IP-based access control list can be implemented on the router to restrict access to SNMP information. A disadvantage to this approach is that it limits the ability to distribute SNMP collection processes to other hosts. Another method to limit the exposure of passwords is to create a management VLAN. SNMP access can be restricted to hosts with IP addresses on the management VLAN. Creating a management VLAN complicates the network architecture and makes it difficult to remotely manage devices.

The telnet protocol faces the same security problems as SNMPv1. The SSH protocol has the authentication, privacy, and access control that telnet lacks, but it is not available on the routers in the testbed. Telnet access to the router can be restricted by creating an access control list. Care should be taken so that the Expect script does not expose sensitive password information to the process list [17].
In this system, an Expect script collects data from routers at an hourly interval. The script captures the ARP cache contents of each router and writes the ARP data to a file. The script organizes ARP files by router name and timestamp. After data collection completes, a separate process reads ARP files and updates the database. Section 2.1.2 gives details of the database design. Section 2.1.3 describes algorithms used to process ARP files and update the database. The remainder of this section focuses on the details of collecting ARP data from routers.

The data collection interval is an important system parameter. The collection process should run at a frequency less than the timeout associated with the data. For Cisco routers, the default ARP cache timeout is 4 hours. The default timeout for the CAM and bridge tables of switch and bridge routers are less than 15 minutes. In the initial ARP system implementation, the router’s ARP, CAM, and bridge tables were collected hourly. The hourly collection interval is sufficient for the ARP table, but the CAM and bridge table data times-out at a faster rate than the ARP entries, so it is not practical to combine data from these tables. The amount of time it takes to process data also impacts the data collection interval. Processing of one data sample should complete before another collection process begins.

Several error conditions can occur during the ARP collection process. For example, if a router interface does not respond when the Expect script is run, no ARP data is available for the collection interval. In this system, no retries are attempted, and the script writes an error message to the ARP file. There may be other errors in the ARP file itself such as an incomplete MAC address, or multiple entries combined on a single line. The ARP processing module handles these error conditions.

2.1.2 ARP Database Tables

This work addresses the problem of reducing large amounts of ARP data into meaningful, manageable chunks. This section presents a database table design for storing data collected from routers. Section 2.1.3 describes algorithms for processing raw ARP files and storing their contents in database tables.
Goals of the database table design include minimizing the size of ARP records, eliminating redundant information, and maximizing the efficiency of data retrieval. The table design is flexible so that it can support new types of devices that may be added to the network.

The Arplog table described in Table 2.1 stores ARP file entries. The Arplog table is similar to a row of data in the router’s ARP cache. A record in the Arplog table contains the IP to MAC address mapping, an identifier used to indicate the interface where the mapping was seen, and dates corresponding to the time period during which the mapping was observed. The total size of an Arplog table record is 52 bytes. Instead of the Oracle date type, character strings are used to represent the first_seen_date and last_seen_date fields. The Oracle date type only uses 7 bytes to represent a date and time, but there is overhead associated with converting a date to and from this compressed format\(^1\). The ADDR_IP field uses 16 bytes so that IPV6 addresses [12] can be stored. IP and MAC addresses are converted from dotted octets to hexadecimal for storage in the table. IP and MAC addresses are stored as hex strings in raw format. This format reduces the space required for storage and avoids conversion overhead associated with the varchar2 data type.

The Xref_Interface table in Table 2.2 stores router interfaces. The IF_ID field uniquely identifies individual interfaces. The two-byte IF_ID field size stores up to 65,535 interfaces. In the current system design, the processing algorithm automatically inserts records into the Xref_Interface table when a new interface is found in an ARP, CAM, or bridge file. The ROUTER_ID associates the interface with a unique identifier assigned to each router.

The Xref_Vlan table in Table 2.3 stores VLAN numbers found in the ARP file of switch-routers. The processing algorithm dynamically adds VLANs as they are found in ARP files.

\(^1\)In retrospect, it may be better to use the Oracle date format. The character string format was used while trying to isolate other database performance issues.
Table 2.1 Arplog Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR_MAC</td>
<td>NOT NULL</td>
<td>RAW(6)</td>
</tr>
<tr>
<td>ADDR_IP</td>
<td>NOT NULL</td>
<td>RAW(16)</td>
</tr>
<tr>
<td>IF_ID</td>
<td>NOT NULL</td>
<td>RAW(2)</td>
</tr>
<tr>
<td>FIRST_SEEN_DATE</td>
<td>NOT NULL</td>
<td>VARCHAR2(14)</td>
</tr>
<tr>
<td>LAST_SEEN_DATE</td>
<td>NOT NULL</td>
<td>VARCHAR2(14)</td>
</tr>
</tbody>
</table>

Table 2.2 Xref_Interface Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF_ID</td>
<td>NOT NULL</td>
<td>RAW(2)</td>
</tr>
<tr>
<td>ROUTER_ID</td>
<td>NOT NULL</td>
<td>RAW(1)</td>
</tr>
<tr>
<td>IF_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>VLAN_ID</td>
<td>NOT NULL</td>
<td>RAW(4)</td>
</tr>
<tr>
<td>IF_DESCRIPTION</td>
<td>NOT NULL</td>
<td>VARCHAR2(100)</td>
</tr>
<tr>
<td>ADDR_MAC</td>
<td></td>
<td>RAW(6)</td>
</tr>
<tr>
<td>CREATE_DATE</td>
<td></td>
<td>DATE</td>
</tr>
<tr>
<td>CHANGE_DATE</td>
<td></td>
<td>DATE</td>
</tr>
</tbody>
</table>

The Xref.Router table, which is described in Table 2.4, stores router information including a unique identifier for each router, the router’s name, IP address, and audit information.

The Xref.Manufacturer table, described in Table 2.5, stores a mapping between MAC address prefixes and the manufacturer name. The data file is available on
the IEEE web site [3]. The file can be downloaded, parsed, and loaded into the database table. Examples of the data stored in the Xref_Manufacturer table appear in Table 2.6.

The Arplog_Updates_History table in Table 2.7 keeps track of updates to all
Table 2.5 Xref_Manufacturer Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC_PREFIX</td>
<td>NOT NULL</td>
<td>RAW(3)</td>
</tr>
<tr>
<td>COMPANY_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(100)</td>
</tr>
</tbody>
</table>

Table 2.6 Xref_Manufacturer Sample Data

<table>
<thead>
<tr>
<th>MAC Prefix</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000A</td>
<td>OMRON TATEISI ELECTRONICS CO.</td>
</tr>
<tr>
<td>00000B</td>
<td>MATRIX CORPORATION</td>
</tr>
<tr>
<td>00000C</td>
<td>CISCO SYSTEMS, INC.</td>
</tr>
<tr>
<td>00000D</td>
<td>FIBRONICS LTD.</td>
</tr>
<tr>
<td>00000E</td>
<td>FUJITSU LIMITED</td>
</tr>
</tbody>
</table>

database tables. Since tables are updated automatically and with different frequencies, it is helpful to store information about the table updates.

The Arplog_Events_History table, described in Table 2.8, records errors detected while processing ARP files. Error conditions that have been identified include missing files, incomplete files, and files containing questionable data. When the processing algorithm detects an error condition, a new record is added to the Arplog_Events_History table. It is helpful to look at the errors for debugging purposes and when determining the usefulness of a specific entry in the Arplog table.

2.1.2.1 Database Indexes and Table Constraints

The choice of database indexes is essential to the system performance. We created an index for each field of the Arplog table. To improve query performance, we stored
### Table 2.7 Arplog_Updates_History Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>ROUTER_ID</td>
<td>NOT NULL</td>
<td>RAW(1)</td>
</tr>
<tr>
<td>UPDATE_DATE</td>
<td>NOT NULL</td>
<td>VARCHAR2(14)</td>
</tr>
</tbody>
</table>

### Table 2.8 Arplog_Events_History Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT_ID</td>
<td>NOT NULL</td>
<td>RAW(1)</td>
</tr>
<tr>
<td>ROUTER_ID</td>
<td>NOT NULL</td>
<td>RAW(1)</td>
</tr>
<tr>
<td>EVENT_COMMENT</td>
<td></td>
<td>VARCHAR2(100)</td>
</tr>
<tr>
<td>EVENT_DATE</td>
<td>NOT NULL</td>
<td>VARCHAR2(14)</td>
</tr>
</tbody>
</table>

indexes in a separate tablespace from the database tables. Since there are no database constraints on the Arplog table, the processing algorithm enforces constraints such as no duplicate records with the same IP address, MAC address, interface identifier, and timestamp. Periodically dropping and creating the Arplog table indexes improves the query performance.

Other tables in the system have primary key constraints and indexes on the constraint fields. Table 2.9 lists the primary keys for each table. Indexes have been created for each constraint.

### 2.1.3 ARP Processing Algorithms

The generalized ARP processing architecture consists of reading the ARP files and error files and updating the database. If the same MAC/IP/interface is seen in
Table 2.9 Database Table Constraints

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Primary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xref_Interface</td>
<td>IF_ID</td>
</tr>
<tr>
<td>Xref_Vlan</td>
<td>VLAN_ID</td>
</tr>
<tr>
<td>Xref_Router</td>
<td>ROUTER_ID</td>
</tr>
<tr>
<td>Xref_Manufacturer</td>
<td>MAC_PREFIX, COMPANY_NAME</td>
</tr>
</tbody>
</table>

consecutive ARP files, the processing algorithm condenses the entries, reducing the total number of Arplog table entries. ARP files contain the following fields: router IP address, timestamp, IP address, MAC address, and interface name. Figure 2.2 contains a description of the ARP processing algorithm for regular routers.

For routers that perform switching between VLANs, two files need to be processed. The ARP file contains the following fields: router IP address, timestamp, IP address, MAC address, and VLAN number. The CAM file contains: router IP address, timestamp, VLAN number, MAC address, and interface name. Data from the two files are combined to determine the switch-router’s port number where a MAC/IP address mapping was seen. Care needs to be taken when combining the data from the two files. There may be MAC addresses in the CAM file that do not appear in the ARP file. There may be more than one IP address in the ARP file for a MAC address found in the CAM file.

Bridging routers are similar to switch-routers in that there are two files that need to be processed and correlated. The ARP file contains the router IP address, timestamp, IP address, MAC address, and bridge virtual interface fields. The Bridge file contains router IP address, timestamp, MAC address, and interface name. Each file may contain a different number of entries. A MAC address from the ARP file can be used to find the interface in the bridge file.
1) Get the max update_date from Arplog_Updates_History
2) Get all open ARP entries for the router
3) For each ARP file entry do
   a. If the MAC address is incomplete, skip it
   b. Convert the MAC and IP addresses to hex
   c. Get the interface ID by the interface name and router_id
   d. If the interface is not found, add it to Xref_Interface
   e. Check if the ARP entry exists in the open list
   f. If the entry exists then update the last_seen_date
      i. Otherwise, insert a new Arplog record
4) If there is an error file in the ARP directory
   a. Open the error file
   c. Insert errors into Arplog_Events_History table
6) Insert a new record in the Arplog_Updates_History
7) If Xref_Interface changed, insert a record in Arplog_Updates_History

Figure 2.2. ARP Processing Algorithm

2.1.4 User Interface

We designed a web-based user interface to test database query performance and to help answer network management questions including:

- Given a device's MAC address and time interval, what IP addresses did the device have?
- Given an IP address and time interval, which devices used the IP address?

Figure 2.3 shows a screen capture of the user interface. The screen capture shows the results of searching for the IP address “132.235.196.158”. The most recent ARP
record is shown on the first row of the search results. The First Seen Date and Last Seen Date fields show that the most recent ARP mapping was seen in consecutive ARP files from 4/7/2003 at 10am until 4/7/2003 at 5pm. The tabs below the search results show other information related to the IP address. The Router tab shows the name and IP address of the router where the data was collected. This tab also displays the router interface and VLAN number. The ARP Log tab shows entries from the Arplog_Updates_History and Arplog_Events_History tables related to the Arplog record. The Manufacturer tab shows the device manufacturer based on the MAC address prefix. The Subnet tab shows information about the subnet that the IP address belongs to.

Answering questions about the physical location of the device and its possible users requires collecting CAM data from switches and using other data sources such as inventory information and directories. Section 2.2 describes the switch data collection system design. Section 2.3 outlines the steps taken to integrate the ARP and CAM systems.

2.2 CAM System Design

The ARP system described in the previous sections is useful for determining the IP address that a device used during some time interval. The ARP system, however, does not have information about the physical location of devices. To obtain this information, it is necessary to find the switch and port where a device is attached to the network. A switch’s CAM table contains MAC address to switch port mappings. This section describes the design of a system for collecting, processing, and storing CAM table data from switches. Section 2.3 shows how inventory information linking a switch and port to a physical location can be combined with the switch CAM table information to locate a device and how this information is integrated into the ARP interface.
2.2.1 CAM Data Collection

A switch’s CAM table contains unicast entries for which the bridge has forwarding information. The switch’s bridging function uses the CAM table to determine how to propagate received frames. A switch’s CAM table contains a mapping of MAC address to port number and a status indicator. The Bridge MIB RFC [11] explains the CAM table MIB objects available through SNMP.

Collecting CAM table data from switches is challenging because there are over 600 switches on the Ohio University network. The default CAM table timeout on
switches is typically five minutes. It is not feasible to collect the CAM tables of 600 switches at a five-minute interval even with fast hardware and a parallel collection process. There is a tradeoff between the accuracy of the data and the CAM table timeout setting. Increasing the CAM table timeout increases the switch’s memory requirements and may have security implications.

2.2.2 CAM Database Tables

The Switch table, described in Table 2.10, stores information about switches including a unique identifier for each switch, the switch’s name, MAC address, IP address, number of ports, model number, and audit information. It is important to keep the Switch table updated, since the table drives the switch data collection and processing.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCH_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SWITCH_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>ADDR_MAC</td>
<td>NOT NULL</td>
<td>RAW(6)</td>
</tr>
<tr>
<td>ADDR_IP</td>
<td>NOT NULL</td>
<td>RAW(16)</td>
</tr>
<tr>
<td>CREATE_BY</td>
<td>NOT NULL</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CREATE_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
<tr>
<td>CHANGE_BY</td>
<td>NOT NULL</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CHANGE_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
<tr>
<td>NUM_PORTS</td>
<td>NOT NULL</td>
<td>NUMBER</td>
</tr>
<tr>
<td>MODEL_TYPE</td>
<td></td>
<td>VARCHAR2(32)</td>
</tr>
</tbody>
</table>

The Cam table, described in Table 2.11, stores CAM data collected from switches. A record in the Cam table contains the MAC address to switch port mapping, and
the time interval during which the mapping was observed. The number data type was used for the SWITCH_ID and PORT fields. The Oracle number data type is a variable size field, so the size of a Cam record depends on the contents of the SWITCH_ID and PORT fields. The average observed size of Cam table records is 26 bytes.

Table 2.11 Cam Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR_MAC</td>
<td>NOT NULL</td>
<td>RAW(6)</td>
</tr>
<tr>
<td>SWITCH_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PORT</td>
<td>NOT NULL</td>
<td>NUMBER</td>
</tr>
<tr>
<td>FIRST_SEEN_DATE</td>
<td>NOT NULL</td>
<td>VARCHAR2(14)</td>
</tr>
<tr>
<td>LAST_SEEN_DATE</td>
<td>NOT NULL</td>
<td>VARCHAR2(14)</td>
</tr>
</tbody>
</table>

The Cam_Updates_History table described in Table 2.12 keeps track of updates to the Cam table. Since it is prohibitive to keep track of individual updates for 600 switches, the Cam_Updates_History table only stores the last time the switch data collection process was successful. A better approach to keeping track of data collection history may be to keep track of cases where data collection fails for a particular switch.

Table 2.12 Cam_Updates_History Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE_DATE</td>
<td>NOT NULL</td>
<td>VARCHAR2(14)</td>
</tr>
</tbody>
</table>
2.2.3 CAM Collection and Processing Algorithms

The switch data collection algorithm uses information about the network topology to produce a list of unique device MAC addresses and the switch and port where the device is connected to the network. Figure 2.4 gives the algorithm used to collect CAM tables from switches. In the switch data collection algorithm, most of the time is spent waiting for the result of SNMP queries. Since the SNMP queries are independent, they can be executed in parallel to reduce the amount of time it takes to collect data from all switches.

1) Create a MAC ignore list. The list consists of MAC addresses for routers and default gateways
2) Get the list of end-point switches. The list includes the IP address, MAC address, and name of the switch.
   a. Add all switch MAC addresses to the MAC ignore list.
3) For each end-point switch do
   a. Get the contents of the Interfaces MIB (1.3.6.1.2.1.2).
      i. If The ifDescr = "Backplane" then add the port number to the port ignore list
   b. Get the contents of the dot1dTpFdbTable (dot1dTpFdbPort, dot1dTpFdbAddress, and dot1dTpFdbStatus) ignoring entries that have status="self" and port 0.
      i. If a port has a MAC address from the MAC ignore list then that port is added to the port ignore list
   c. For each port not on the port ignore list do
      i. Print the MAC address and port number

Figure 2.4. Switch Data Collection Algorithm
The database loading process reads the CAM file created for each switch and updates the Cam database table. The data aggregation problem for router ARP entries was solved by updating the last_seen_date of an Arplog entry if the same IP to MAC address mapping was seen in consecutive samples. The data aggregation problem is more difficult for switches because it is not possible to collect data at a frequency less than the CAM table timeout. The database loading algorithm increases the aggregation granularity for switch data. Instead of the hourly aggregation window used by the ARP processing algorithm, switch data is rolled up into day intervals. If a MAC address to switch port mapping was seen in consecutive days, the last_seen_date of the Cam record is updated. If a new MAC address or MAC/switch/port mapping is seen, a new Cam record is created with the first_seen_date and last_seen_date set to the current day. Details of the switch data loading algorithm are given in Figure 2.5.

1) Get the list of switches
2) Get a cache of Cam table entries from the current and previous days
3) For each switch do
   a. Open the CAM file for the switch
   b. For each MAC/switch/port in the CAM file do
      i. If the MAC/switch/port is found in the cache, update the last_seen_date
      ii. Otherwise, create a Cam record for the MAC/switch/port with and first_seen_date=last_seen_date=current day
4) Insert a record into the Cam_Updates_History table for the current timestamp

Figure 2.5. Switch Data Processing Algorithm
2.3 ARP and CAM Integration

In order for the CAM system to effectively find the physical location of a device, a list of switches and an accurate mapping of switch ports to physical locations is required. When a new switch is added to the network, it should also be added to the database so that the new switch’s CAM table can be collected. Similarly, changes to wiring between a switch port and jack should be maintained. If the list of end-point switches is not accurate, MAC addresses will be associated with the downlink port of another switch. If the mapping of switch port to physical location is inaccurate, the system will give the wrong physical location for the device.

At Ohio University, the switch port to physical location mapping was maintained in a flat file. A database system called NetOhio was implemented to maintain the data and integrate with other NMI components. Open source network inventory management systems such as LANdb [18] could be used instead of the custom system that was developed.

In order to find the possible users of a device, a directory can be used to link people to physical locations. The usefulness of this approach depends on the accuracy of the directory information. Ohio University maintains an electronic directory containing the address and contact information for students, faculty, and staff. When a device is located using the CAM system, a query can be made to find people associated with the physical location.

Using a directory to identify the users of a device based on the device’s physical location is a limited approach. Users may have access to a device, but may not be associated with the device’s physical location. Other data sources such as authentication logs may be used to associate a user with an IP address during some time interval.

The Location tab pictured in Figure 2.6 shows the results of integrating ARP, CAM, and physical location information. Starting with an IP address, the ARP system gives a device’s MAC address. The CAM system gives the switch and port
where a MAC address was seen. Records in the Arplog and Cam tables are tied together by the first seen date and last seen date. The time granularity between the Arplog and Cam tables is different, so Cam table entries for the given day are displayed. The NetOhio database maps the switch and port to the building, room, jack, and jack position. The building address and room number can then be used to search for people in the directory.

Figure 2.6. Screen Capture Showing Location Information

2.4 Test Environment

Ohio University has a “No Single Point of Failure” (NSPOF) architecture for all production elements of both the network and the central computing and storage environments. This architecture was used to evaluate the system performance. The key elements of the test environment are described below.

2.4.1 Network Topology

The backbone of Ohio University’s network is depicted in Figure 2.7. Two Cisco 6500 switch routers act as the core of Ohio University’s network. The core is extended by a Cisco 5500 switch router and a Cisco 4000 serving edges of the network.
Ohio University Network Backbone
Overview

- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking
- GigEth - VLAN trunking

Figure 2.7. Ohio University Network Topology Diagram
The network border is operated by two Cisco 7200 routers. Each of the 7200s is responsible for the connectivity to one of the two ISPs that service the University. Connectivity between the core and the border is designed as a NSPOF system. The University maintains two Internet I links (OARnet and Genuity) and one Internet II link (OARnet). Internet I and Internet II service from OARnet is provided on a single ATM connection. The border routers are responsible for Quality of Service (QoS) tagging and queuing.

Groups of buildings on the Athens campus are tied together with a switched LAN, and that LAN is connected into the core routers. Some designation is made between Residential buildings and the Academic/Administrative buildings. Connections are being developed so that each of these networks is redundantly connected to each of the 6500 core routers. This will allow the core and the network interconnects to operate in a NSPOF manner.

One internal router is designated to operate the T1 data connections to the Ohio University regional campuses and a bypass link to one of the local ISPs. Another internal router connects the University’s DSL customers. Plans for a campus wide wireless initiative are being developed. One of the requirements of this initiative is the need to authenticate the users. The authentication infrastructure that has been chosen requires a middlebox access control gateway. The gateway that has been selected functions as a router. One of these middleboxes will be located at the head-end of each of the residential and academic/administrative LANs.

Ohio University has a fully switched network with a switch port dedicated to each end-point node. There are 600 Layer 2 switches deployed representing different generations of Enterasys hardware.

2.4.2 Database Infrastructure

We used Oracle as the back-end data store for the ARP/CAM system. The database instance runs within the NSPOF architecture on Compaq Tru64 5.x Tru-Clusters. The choice of Oracle versus open source databases such as MySQL has
traditionally been supported by the immaturity of the open-source varieties. They have lacked vital features like table and row level locking, advanced recovery methods, support for complex SQL statements like subqueries, and set operators.

We used Oracle Forms to develop the user interface. Oracle Forms is a rapid application development environment that is part of the Oracle 9i Application Server. We chose Oracle Forms as the tool for designing the interface because a prototype of the application could be developed without much programming effort. Moreover, Oracle Forms can be deployed on the Internet through Oracle's Apache Web Server. Oracle Forms is closely integrated with the Oracle database, so the user interface has optimal query performance. Although Oracle Forms is a java-based technology, it is only available for Microsoft Windows platforms.

2.4.3 Clustered Computing Environment

This project uses a cluster consisting of 2 Compaq DS20 (2 processors each @ 800 MHZ) and 2 single processor DS10 computers. This cluster will support 2 tiers of a 3 tier Oracle infrastructure — the Oracle database and the Oracle 9i applications environment. The third tier of the Oracle environment will be the client running a web browser or other Oracle clients such as native SQL tools.

2.4.4 Storage Area Network

A Compaq StorageWorks SAN, consisting of parallel sets of Brocade Fibre Channel switches connecting the host computers with two existing arrays with 2 Terabytes of current capacity. The arrays include an EMA12000 and an ESA12000 array with fully redundant controllers and cache. The use of the SAN allows easy expansion of storage in a very reliable and high performance manner and leverages existing management and backup strategies in an incremental fashion.
3. RESULTS

This chapter gives a detailed analysis of the ARP/CAM system described in Chapter 2. During system development, we performed several experiments to determine appropriate data collection methods, polling intervals, and data aggregation methods. The remaining sections explain the experiments and analyze the results.

3.1 Experimental Results and Analysis

During the development process, we performed several experiments to determine the effectiveness of data collection, processing, storage, and retrieval. We performed similar experiments for the ARP and CAM systems. The results of these experiments are discussed in this section. This section also describes implementation problems encountered during system development.

3.1.1 Router Data Collection and Processing

The ARP collection scripts capture the ARP cache contents of 15 routers in approximately 3 minutes. The data collection scripts do not significantly impact the router’s CPU performance. Figure 3.1 shows the CPU utilization for one of the most highly utilized campus routers. The graph shows small spikes in the router’s CPU utilization during the hourly data collection period.

The ARP data collection system was not tested in a situation where the router’s CPU was fully utilized. If the data collection script fails to collect data from a router for a specific time-stamp, no data is available, and an error is recorded in the database. The data collection system could be improved by performing retries if the router does not respond initially.

While developing the ARP system, raw ARP files collected from routers were
stored for debugging purposes. As the system grew in size, it became impractical to store the raw ARP files on disk. One month of uncompressed ARP files collected from the 15 campus routers consumed 2 GB of disk space. Approximately 24 GB of disk space is required to store ARP files for one year. Compressing the data leads to a significant space savings. One month of ARP data can be compressed to 280 MB. If compression is used, 3.4 GB of disk space can store raw ARP data for one year. Compressed ARP files are archived on CD for backup purposes. It is sufficient to use the database as the authoritative source of ARP data, and to perform periodic backups of the database.

We found the hourly data collection interval to be appropriate for collecting router ARP tables. Since the routers had a four-hour ARP table timeout, data was not lost in the collection process. A side-effect to sampling ARP data at a higher frequency than the timeout was that an IP/MAC pair remains in the router’s ARP cache for several sampling cycles before it is removed. A laptop can be moved from one location to another, but in the ARP system, it would appear as though it were in two locations at the same time. This fact makes it difficult to interpret the search results when trying to location a device through the user interface.

Another case where the router’s ARP cache gives misleading information is when there is a network device that performs ARP for other devices. In this case, the client’s MAC address appears in the router’s ARP cache with the IP address of the
device performing ARP for the client. A client’s MAC address can be traced to the wireless access point where it is attached to the network using CAM data collected from switches, but the IP address of a wireless client is not found in the ARP data collected from routers.

The ARP processing module was implemented in Perl because of its text processing capabilities and associative arrays which were used to merge the contents of different files. Since Perl code is interpreted, it is less efficient than compiled code. The Perl database independent interface (DBI) module was used to perform database queries. The sqlplus command was used on temporary files to perform database update and insert operations.

Initially, the ARP processing module took more than one hour to process ARP files from all campus routers. This problem was caused by database operations performed during file processing. After reading a line from the ARP file, the processing module performed a database lookup and an insert or update depending on the result.

To improve the processing module’s performance, open records from the Arplog database table were cached in memory before the ARP file was processed. Instead of performing database updates and inserts for each line in the ARP file, SQL statements were written to a temporary file. After ARP file processing finished, the database commands were executed. Caching open Arplog records and separating the database operations from file processing reduced the amount of time it took to process all ARP files from over one hour to approximately 3 minutes. The processing module with caching code did not impose a significant CPU load. The memory requirements for the caching algorithm depend on the number of open Arplog entries, but the observed memory utilization was negligible.

Tables 3.1 and 3.2 show the effects of caching open Arplog records from the database. The caching code was originally included in the CAM processing algorithm because the switch-routers had the most ARP cache entries. The Update Time is the time that it takes to run the SQL commands. This time is a constant that
depends on the number of update and insert statements in the SQL file.

Table 3.1 Results of Caching for CAM Routers

<table>
<thead>
<tr>
<th>Router</th>
<th>Type</th>
<th>Processing Time (s)</th>
<th>Update Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_parnet</td>
<td>arp</td>
<td>11.888</td>
<td>0.26</td>
</tr>
<tr>
<td>6500a</td>
<td>cam</td>
<td>9.629</td>
<td>74.558</td>
</tr>
<tr>
<td>6500b</td>
<td>cam</td>
<td>9.262</td>
<td>66.893</td>
</tr>
<tr>
<td>airport</td>
<td>arp</td>
<td>41.571</td>
<td>0.146</td>
</tr>
<tr>
<td>dsl</td>
<td>bridge</td>
<td>405.984</td>
<td>1.528</td>
</tr>
<tr>
<td>ouecu</td>
<td>arp</td>
<td>23.876</td>
<td>0.144</td>
</tr>
<tr>
<td>regionals</td>
<td>arp</td>
<td>121.565</td>
<td>0.144</td>
</tr>
<tr>
<td>ridges</td>
<td>cam</td>
<td>2.787</td>
<td>3.29</td>
</tr>
<tr>
<td>chillicothe</td>
<td>arp</td>
<td>216.702</td>
<td>0.151</td>
</tr>
<tr>
<td>eastern</td>
<td>arp</td>
<td>159.115</td>
<td>0.144</td>
</tr>
<tr>
<td>lancaster</td>
<td>arp</td>
<td>410.563</td>
<td>0.144</td>
</tr>
<tr>
<td>pickerington</td>
<td>arp</td>
<td>23.572</td>
<td>0.162</td>
</tr>
<tr>
<td>proctorville</td>
<td>arp</td>
<td>29.714</td>
<td>0.144</td>
</tr>
<tr>
<td>southern</td>
<td>arp</td>
<td>291.341</td>
<td>0.152</td>
</tr>
<tr>
<td>zanesville</td>
<td>arp</td>
<td>333.057</td>
<td>0.145</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>2078.738</strong></td>
<td><strong>147.745</strong></td>
</tr>
</tbody>
</table>

The goal of the initial ARP system design was to capture as much detail as possible from router data. In the absence of CAM data collected from end-point switches, the CAM and bridge data collected from routers provided more detail about the location of a device and the VLAN or port where the device was attached to the network. The most significant problem in the ARP system design was determining how to combine CAM and bridge data that had a five-minute timeout with ARP data that
Table 3.2 Results of Caching for All Routers

<table>
<thead>
<tr>
<th>Router</th>
<th>Type</th>
<th>Processing Time (s)</th>
<th>Update Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>borderoarnet</td>
<td>arp</td>
<td>3.02</td>
<td>0.197</td>
</tr>
<tr>
<td>6500a</td>
<td>cam</td>
<td>9.164</td>
<td>63.668</td>
</tr>
<tr>
<td>6500b</td>
<td>cam</td>
<td>9.073</td>
<td>60.186</td>
</tr>
<tr>
<td>airport</td>
<td>arp</td>
<td>2.555</td>
<td>0.214</td>
</tr>
<tr>
<td>dsl</td>
<td>bridge</td>
<td>2.681</td>
<td>1.436</td>
</tr>
<tr>
<td>ouecu</td>
<td>arp</td>
<td>2.556</td>
<td>0.185</td>
</tr>
<tr>
<td>regionals</td>
<td>arp</td>
<td>2.531</td>
<td>0.346</td>
</tr>
<tr>
<td>ridges</td>
<td>cam</td>
<td>2.811</td>
<td>3.072</td>
</tr>
<tr>
<td>chillicothe</td>
<td>arp</td>
<td>2.531</td>
<td>0.355</td>
</tr>
<tr>
<td>eastern</td>
<td>arp</td>
<td>2.507</td>
<td>0.352</td>
</tr>
<tr>
<td>lancaster</td>
<td>arp</td>
<td>2.548</td>
<td>0.548</td>
</tr>
<tr>
<td>pickerington</td>
<td>arp</td>
<td>2.504</td>
<td>0.193</td>
</tr>
<tr>
<td>proctorville</td>
<td>arp</td>
<td>2.486</td>
<td>0.166</td>
</tr>
<tr>
<td>southern</td>
<td>arp</td>
<td>2.531</td>
<td>0.451</td>
</tr>
<tr>
<td>zanesville</td>
<td>arp</td>
<td>2.555</td>
<td>0.581</td>
</tr>
</tbody>
</table>

Total                                52.053       131.95

had a four-hour timeout. In the algorithm for combining the ARP data with CAM or bridge data, if a MAC address from the ARP table was not found in the CAM or bridge table, it was not possible to determine the port where the device was seen, and the port was undefined. If a MAC address from the CAM or bridge table was not seen in the ARP table, the IP address of the Arplog record was set to the router’s IP address to preserve the fact that the MAC address was seen by the router.

The algorithm that combined ARP and CAM/bridge data caused problems with
the ARP data roll-up\textsuperscript{1}. If the same IP/MAC/interface mapping was seen in consecutive samples, we incremented the last\_seen\_date instead of creating a new Arplog table entry. The CAM and bridge processing algorithms produced records that oscillated between having a known and unknown interface name. A similar problem occurred with the MAC addresses that were assigned the router’s IP address. Because the exact same IP/MAC/interface mapping was not seen in consecutive samples, hundreds of unnecessary Arplog records were created. The unnecessary Arplog records slowed database query performance and made the search results difficult to interpret.

An improved design for the ARP system would not combine the ARP table with the CAM or bridge tables. Instead, the CAM system should handle data collection and processing of the switching and bridging functions of the router. If this approach is used, the CAM collection algorithm would eliminate most of switch-router’s CAM table entries because they are down-link ports to other switches. Only devices attached directly to the switch router would be discovered. The ARP system would have fewer Arplog entries, and the query performance of the user interface would improve.

Figure 3.2 shows the growth of total Arplog table entries from April 2002 to March 2003. During this time period, approximately 8 million records were added to the Arplog table. The Arplog growth statistics reflect IP address usage patterns at Ohio University. During peak network usage from September to November and from January to June, an average of 1.2 million entries were added to the Arplog table each month. During the off-peak months of December, July, and August, an average of 430,000 new records were added each month.

Storing ARP data in the database offers a significant space savings over the raw data. To store one month of Arplog records in the database requires 3.4 MB. Even if compression is used, the raw ARP data for one month uses 280 MB of space. If the

\textsuperscript{1}If we see the same IP/MAC/Interface mapping in consecutive ARP samples, we update the last\_seen\_date of the existing database entry.
size of database indexes is taken into account, the average size of an Arplog record grows to from 52 to 176 bytes, and the amount of space required to store one month of indexed Arplog records is 114 MB.

![New Arplog Entries By Month](image_url)

**Figure 3.2. New Arplog Table Entries By Month**

### 3.1.2 Switch Data Collection and Processing

The switch data collection and processing algorithms were also implemented using Perl. The Perl Object Environment (POE) was used to manage parallel SNMP query processes. POE [8] is a framework for creating multitasking programs in Perl. The net-snmp Perl module was used to perform SNMP queries.

Using 50 concurrent processes, we collected CAM tables from 595 switches in 6 minutes. Because switch ports are pruned\(^2\) during the collection process, the resulting data contains only end-point MAC addresses. One data sample from all switches requires 6 MB of disk space (576 MB/day if the sampling period is 15 minutes).

\(^2\)The collection process eliminates ports that connect to other switches.
Because of the high volume of data, it is not practical to store CAM data on disk. After processing has completed, the raw CAM files are deleted. Processing and storing the data in the database takes less than 3 minutes.

CAM table collection did not have a noticeable effect on switch CPU utilization. The switches in Ohio University’s network have separate CPUs for management and switching functions, so SNMP queries do not impact the main switch CPU load. We observed the management CPU load to be less than 1% during the switch data collection process. If the load on the switch’s management CPU reaches 100%, SNMP packets may be lost. For switches that use a single CPU for both switching and management functions, SNMP queries may affect the CPU load.

We performed experiments to determine a suitable data collection interval. The default CAM table timeout for most campus switches was 5 minutes, but experiments showed that it was not feasible to collect data at an interval less than 15 minutes. The 15 minute collection interval was found to be a good trade-off between system stability, accuracy of data, and the load placed on the host sending SNMP queries.

Some data is lost when using the 15 minute collection interval because of the switch’s 5-minute CAM table timeout. In order to make strong statements about the location of a particular device, the collection interval should be less than the switch’s CAM table timeout.

There are security implications to collecting switch data at a fixed interval. A malicious user could spoof another user’s MAC address near the collection interval. As a result, the malicious user’s real MAC address would not appear in the switch’s CAM table, making the malicious user invisible to the ARP/CAM system. A solution to this problem would be to trigger the switch data collection process at a random time within the collection interval. We did not implement a random collection interval, but future work should consider this security improvement.

The collection and analysis of network topology information can help troubleshoot network problems and help determine which switch ports have end-point MAC ad-
addresses. If topology information is not collected periodically, changes in the network topology or configuration of switches can cause the pruning algorithm to produce inaccurate results. Experiments have shown that the contents of a switch’s CAM table are highly dynamic. If the MAC address of a down-link switch does not appear in a particular sample, a port on the up-link switch may not be eliminated causing the same MAC addresses to appear on two different switches. To avoid this problem, topology discovery should be decoupled from the data collection process.

The data processing algorithm used for switch CAM data faced a similar problem as the ARP system in that the volatility of CAM data complicates the roll-up process. To reduce the number of entries in the Cam database table, the roll-up window was chosen to be one day. A new Cam record is created each time a new MAC/switch/port combination is seen. If the same combination was seen the previous day, then the last_seen_date of the old record is incremented. No database operations are performed if the same MAC/switch/port is seen during the same day.

Duplicate MAC addresses are considered errors in the system, since duplicate MAC addresses that appear on the same LAN can cause network problems for all devices that share the address. If duplicate MAC addresses are found in the collected CAM data, they are not inserted in the database by the processing algorithm. The duplicate MAC addresses are printed to an error file for analysis. An example of a MAC address that was seen on several different LANs at the same time is 02:03:8A:00:00:11. The MAC address belongs to an AOL adapter. Duplicate MAC addresses with the Microsoft manufacturer prefix 00:50:F2 were also seen on different switches. These devices are XBoxes that have user-configurable MAC addresses. Location information from the CAM system can be used to contact users of the misconfigured devices and correct the problems.

We identified one case where duplicate MAC addresses are acceptable. The wireless middleboxes have the same virtual MAC address to allow for roaming between middleboxes. Since this case is not an error, we added the middlebox’s MAC address
to the pruning algorithm's ignore list.

A misconfigured switch was found when duplicate MAC addresses appeared on two connected switches. The collection algorithm could not prune a down-link port because the switch's CAM table timeout was set to 15 seconds. Because of the short timeout, the down-link switch's MAC address did not always appear in the CAM table.

3.1.3 Data Presentation and Query Performance

The ARP/CAM interface proved to be an effective network management tool for answering questions about the past and present location of a network device. Ad hoc database queries can be used to mine the database and produce reports about IP address usage and allocation. These ad hoc queries are useful for network planning. For example, IP address usage for some time period can be determined for a subnet. If the address space is fully utilized, the subnet can be expanded to provide more addresses.

A good example of the tool's usefulness for real-time incident response was during a denial of service attack caused by the SQL Slammer worm [2]. The worm flooded the network with packets while trying to find vulnerable SQL servers. The ARP/CAM interface was used to find switch ports where infected devices were connected. The switch ports were disabled, restoring stability to the network without disabling service to large groups of users. Using the ARP/CAM tool to locate infected devices saved time over the manual process of examining log files from several data sources.

The ARP interface was not useful for answering network planning questions, but specific ad hoc queries were created to find IP address usage for a subnet or router interface. This information was used to manage IP address pool allocation. Another graphical interface was developed to display IP address usage by subnet. Other ad hoc queries are being explored to automate incident response systems.

The average query time for the ARP/CAM interface is 14 seconds. The query time is affected by the number of Arplog records and the database indexes. Figure 3.3
shows the average monthly query time for the ARP interface. Rebuilding database indexes significantly improves the query time. The process of rebuilding indexes was performed manually and took 30 minutes to execute. The process was executed when query times exceeded 20 seconds. The query time is trending upward, but it is due to the large number of unnecessary ARP records inserted in the database. Section 4.3 proposes a new table structure that bounds the number of Arplog table records and improves semantics of the data. The total number of Arplog records could be reduced by off-loading older Arplog records from the database.

![Average Arplog Table Query Time](image)

*Figure 3.3. Average Query Time By Month*
4. CONCLUSIONS

This work studied the feasibility of tracking the physical location of network devices over time using data collected from routers and switches. In this chapter, we summarize the experimental results and analysis of the ARP and CAM systems and provide directions for future work. We begin this chapter by outlining the contributions of this work and comparing this system to existing network management systems.

4.1 Related Work

This work focuses on the design and analysis of a system for tracking the historical location of devices on the network. The outcomes of this work include

- A system for tracking the location of network devices
- An analysis of key system parameters and recommended settings for parameter values
- An analysis of space requirements for storing large volumes of historical data
- A description of several uses for the historical data
- A generalized architecture for collecting, storing, and retrieving network metadata

Our system differs from existing network management software because it presents a generalized model for collecting and aggregating historical network metadata that can be extended beyond the scope of router ARP tables and switch CAM tables.
We performed experiments to find the limits of collection intervals, data aggregation levels, and space requirements for storing historical data. In the following sections we compare our work to existing network management systems including NetReg/NetMon, Spectrum, and OpenView.

4.1.1 NetReg/NetMon


The main goal of NetReg/NetMon is to keep track of MAC address registrations and enforce network access policy associated with these registrations. MAC address registration does not imply attributable network access. A malicious user could spoof the MAC address of a registered device to gain unauthorized access to the network. Since Ohio University’s network does not require MAC address registration, the system that we built does not track associations between devices and users. In our system, we determine a device’s users by the physical location of the device.

NetMon stores the CAM tables from all switches on the network. In a typical network, if two hosts communicate with one another, the CAM tables of all switches between them are updated to reflect the communication path. The MAC Addresses of the hosts appear on the uplink ports of infrastructure switches. To find the endpoint port of a device, NetMon looks at all ports where a given MAC address was seen. The port that has the fewest number of other MAC addresses is the assumed location.

Our experiments showed that the least number of neighbors algorithm used by NetReg/NetMon to locate a device is not deterministic, and it is less efficient than the
deterministic algorithm that uses topology information to find the end-point ports of all network devices. The least number of neighbors algorithm does not identify duplicate MAC addresses on the network. The NetReg/NetMon system assumes all MAC addresses on the network are unique, but there are valid cases where the same MAC address can appear on multiple switch ports. One example is wireless middleboxes that use the same MAC address to support roaming between network segments. In other cases, duplicate MAC addresses within the same subnet can cause errors, so it is necessary to locate all devices that share the same MAC address.

NetMon may not scale to collect CAM data from a large network with many switches. NetMon uses two parallel processes for data collection. One process collects data from slower devices, and the other collects from faster devices. The Ohio University network contains more than 600 switches. Collecting the CAM tables from these switches takes an average of ten seconds per switch, but fifty of the switches have a collection time exceeding twenty seconds. The CAM table collection time for all switches could exceed one hour if only two parallel processes are used.

4.1.2 Spectrum

Aprisma’s Spectrum network management software monitors a network’s health and performance [1]. Spectrum is available in three versions for managing networks of different sizes. Ohio University uses Spectrum for managing switch configurations, maintaining network topology maps, and managing network faults. Spectrum’s key features include

- Event management (alarms, fault isolation and root cause analysis)
- Discovery and mapping of network devices
- Polling
- Reporting
- Trap Handling
• LAN/WAN management

Spectrum has a configurable autodiscovery feature for finding all devices on the network. The topology information can be organized into topology maps that show interconnections between network devices. Spectrum can poll specific device MIBs at different time intervals. This information can be logged and used to calculate performance information for devices. Spectrum’s event management features monitor the network for faults and generate alarms when faults occur.

Spectrum’s MAC Address Locator Tool (MALT) can locate a device on the network given a device’s MAC or IP address. MALT functionality can be accessed through a web-based GUI, a command line interface (CLI), and an API. We tested the CLI and found that the results were often inaccurate, and the response time for location queries was slow. There were many cases where MALT was unable to locate devices connected to the network. The accuracy of MAC address lookups through MALT depends on Spectrum’s topology discovery. Spectrum periodically discovers new nodes by sending the switch’s read-only community string to all nodes on the network. This method of topology discovery is both time consuming and insecure.

4.1.3 OpenView

HP OpenView Network Node Manager (NNM), like Spectrum, is a broad family of network and system management products [4]. NNM provides the following features:

• Discovery and mapping
• Monitoring
• Notification processing
• Reporting
• Data warehousing
• Configuration management
NNM supports a proactive approach to network management. NNM finds the network’s current state by finding what devices are present, how the devices are configured, how devices are performing, and if there are any problems. NNM does this by polling for network information and screening problems through event correlation. NNM collects historical information to identify trends. Future problems can be predicted by analyzing historical information.

NNM automatically discovers network topology and builds a graphical network topology map. The discovery processes uses SNMP-based polling and ICMP requests. NNM continuously polls for the status of devices, network topology changes, and configuration changes.

NNM can monitor critical network devices and send notification when faults occur. When a device fails, NNM’s event correlation evaluates events to pinpoint the cause of a failure. NNM’s predictive features can predict future trouble spots before a failure occurs.

NNM’s data warehouse is a repository of trend, event, and topology data stored in a relational database. NNM has an embedded relational database, but it also has support for external Oracle and Microsoft SQL Sever databases. Archived data can be used for troubleshooting, reporting, and analyzing the network.

4.2 Summary

We developed a system to trace the physical location of network devices by collecting data from routers and switches. The system collects and processes ARP data from routers and CAM data from switches. The router’s ARP table maps IP addresses to MAC addresses, and the switch’s CAM table maps MAC addresses to switch ports. We combined ARP and CAM information to find the switch port where a device is attached to the network. Network inventory information helped identify the building and room serviced by the switch port. We identified a device’s possible users by searching through the campus directory by building name and room number. The
ARP and CAM tables only retain data for a short period of time. One of the goals of this work was to create a temporal database to capture the location of devices and this data for several years. Older data can be used to find network usage patterns and to plan IP address allocation. Recent ARP and CAM information can be integrated with intrusion detection systems to locate devices that are the source or target of a network attack. Users of the compromised device can be contacted, or the switch port can be disabled to remove the affected device from the network.

We developed and tested the ARP/CAM system on Ohio University’s network. The network consists of 15 routers and 600 switches. Router data collection and processing occurred once per hour and took six minutes to complete. We chose the data collection interval for routers and switches to optimize the accuracy of the data while minimizing the overhead of data collection and processing. Since router ARP entries persist for four hours, we determined that an hourly collection interval was appropriate. Switch CAM entries remain in the table for five minutes, however, it is not feasible to collect and process the data at a five minute rate. Experiments determined that a fifteen-minute data collection and processing interval could be achieved for switches.

We designed a graphical user interface to facilitate database searches. Given an IP or MAC address, the interface combines ARP, CAM, inventory, and directory data to locate the device and identify its possible users. The query performance of the interface is fourteen seconds with approximately eighteen million Arplog records in the database. Periodically rebuilding database table indexes reduced the query time.

The experimental results show that the system successfully scaled to collect data from 15 routers and 600 switches. Increasing the number of routers or switches adds a small constant to the collection time, but the impact is not significant. Data processing may not scale as the number of routers and switches increase. Section 4.3 outlines several improvements to data collection and processing that address these scalability issues. The database query performance was negatively impacted by un-
necessary records added by the processing algorithm; however, periodically rebuilding
the database indexes reduced the query time below fourteen seconds. Improvements
to the database table structure and algorithms that populate the table would reduce
the number of spurious records and improve database performance.

4.3 Future Work

This section outlines improvements to the ARP/CAM system design presented in
Chapter 2. Improvements can be made to data collection, processing, and presenta-
tion modules.

The processes that collect data from routers and switches transmit sensitive pass-
word information in plain-text. Even if the password gives restricted, read-only access
to the device, it is dangerous to expose the password and data stream on the network.
Unreliable, connectionless protocols such as SNMPv1 are susceptible to Modification
of Information, Masquerade, Disclosure, and Message Stream Modification attacks [6].
One solution is to create access control lists (ACLs) on the routers and switches to
limit the hosts that collect data. There are management costs to maintaining these
lists. If tools are not available to manage router and switch configurations, the ACLs
need to be maintained by hand. Another way to reduce the exposure of passwords is to
create a management VLAN and disable SNMP access to hosts outside of the VLAN.
Each switch would have a port on the management VLAN. The host performing data
collection would also have an interface on the VLAN. Creating a management VLAN
can complicate the network architecture and make it difficult to remotely manage
switches.

Experiments determined the 15 minute switch data collection interval was a good
trade-off between system stability, accuracy of data, and the load placed on the host
sending SNMP queries, however, some data is lost, since the switch’s CAM table
entries time-out after five minutes. In order to make strong statements about the
location of a particular device, the collection interval should be less than the switch’s
CAM table timeout.

The switch’s CAM table timeout can be increased to improve the accuracy of collected data. Experiments should be performed to determine if there are security or performance implications to increasing the switch’s CAM table timeout. Sampling theory suggests that a signal should be sampled at greater than twice the maximum frequency. Since sampling can be performed at a frequency of 15 minutes, a CAM table timeout of 30 minutes would improve the accuracy of CAM table data.

The switch data collection process relies on topology information to prune up-link and down-link ports from collected CAM tables. During system development, the switch collection process used static information including the default gateway MAC address and the switch host port MAC address. Because this information is dynamic, it should be gathered independent of the switch CAM collection process. If the MAC address of a down-link switch does not appear on a port for a particular sample, the port cannot be pruned, so all MAC addresses on that port are added to the database even though those MAC addresses belong to the down-link switch. Separating CAM collection from topology discovery will reduce these transient errors. The topology information gathered from switches should include:

- The default gateway IP address
- The MAC address of the switch host data port
- The interface table to determine backplane ports

Changes to the database structure and the ARP processing algorithm could eliminate unnecessary Arplog records, which would improve semantics of the data and reduce query time. In the original system design, ARP and CAM/bridge data collected from routers were combined. New records were added to the Arplog table as new IP/MAC/interface mappings were seen in ARP files. If the same mapping was seen in consecutive samples, the last seen date of an existing Arplog record was
incremented. The CAM/bridge processing algorithms produced records that oscillated between having a known and unknown interface name because the exact same IP/MAC/interface mapping was not seen in consecutive samples. Since CAM data collected from switches gives the physical location of a device, combining the router’s ARP and CAM/bridge tables is unnecessary.

The ARP protocol is not connection-oriented, but aggregated ARP data in the Arplog table could be misinterpreted as a time period during which a device was communicating on the network. Other data sources such as Netflows [14] should be used to determine when the device was sending packets. Routers have a four-hour timeout, so a device that is active on the network for a brief period of time will have an Arplog record that spans the four-hour time period.

For certain applications it may be necessary to obtain the current IP/MAC binding and location of a device. Because we collect the contents of a router’s ARP cache at an hourly interval, there is a delay between the time that an ARP entry is added to the router’s ARP cache and the time that the same binding appears in the database. The current ARP collection process can be adapted to handle asynchronous requests for MAC/IP bindings. A similar problem exists for switches. Ideally, we would like to capture all MAC addresses that appear on the switches. Our experiments showed that it is not possible to collect switch data at an interval short enough to capture all MAC addresses, but the current CAM collection process can be modified to handle asynchronous requests for the location of a device. We could use an algorithm that uses topology information to trace a MAC address to the end-point switch port. We did not implement a mechanism for handling asynchronous location requests, so this feature should be considered in future work.

The Arplog database table could be improved to reflect the fact that an IP to MAC address mapping was seen in a particular sample rather than during a time interval. This could be achieved by replacing the first_seen_date and last_seen_date fields of the Arplog record with a single date field and adding a bit-map with the
number of bits determined by the data collection interval. With an hourly collection interval, the bit-map would contain 24 bits. The processing algorithm could keep track of Arplog records for a particular day and record the fact that an IP address binding was seen in the in a particular sample by setting the bit that corresponds to the sample number. A similar change can be made to the Cam database table, but the bit-map would contain 96 bits.

Using the bit-map approach, the total number of records added to the Arplog table is bounded by the total number of unique IP address bindings seen in a day. More Arplog records may be created with the bit-map approach than with the aggregation method, but no connection-oriented information is implied by the bit-map. Also, it is easier to partition the data, since records do not span more than one day.

Other database improvements could be made to improve database loading and query performance. In the current system, a single database table is used for storing all ARP data. To improve loading performance, a temporary table could be used to store inserted or updated records. After processing completes, the records in the temporary table could be merged with the Arplog table. Query performance can be improved by automating the process of rebuilding Arplog table indexes.

Experiments should be performed to determine if implementing the processing code in a compiled language such as C or C++ would offer a performance gain over the Perl implementation. The Oracle Call Interface (OCI) for C/C++ used to communicate with the database may be more efficient than the Perl DBI libraries.

We used an Oracle database to store processed ARP and CAM data. We chose Oracle because we could leverage existing IT infrastructure and experience. A review of other database engines such as MySQL is warranted to see if it is a viable alternative as the data store for the ARP/CAM system. Additionally, platform-independent tools should be considered for developing the user interface.
BIBLIOGRAPHY
BIBLIOGRAPHY


