Automated Service Discovery for Enterprise Network Management

William Tu, Priya Thangaraj, Jui-hao Chiang
Professor Tzi-cker Chiueh
CEWIT Stony Brook University

March 8, 2009

1 Introductin

A key pillar of enterprise or data center network management is the ability to provide real-time visibility of the configuration and traffic load of the networks under management. More specifically, in general three types of configurational information is of interest to network management tools:

1. Link-layer topology: This shows how switches, routers, hosts and servers are physically connected through Layer-2 links.

2. Network-layer topology: This shows how routers partition hosts and servers into distinct Layer-2 subnets.

3. Network service map: This shows the map between network services and servers, and which hosts access which services.

A networks link-layer topology is a superset of its network-layer topology. However, it is generally much easier to reconstruct the network-layer topology of a network than its link-layer topology, primarily because routers exchange among themselves the routing table information but switches dont. An obvious application of network topology information is to provide a context for displaying and analyzing traffic load data. Another application is to support impact analysis, e.g. what services will be disrupted if a network device such as a switch is pulled off for maintenance. However, such impact analysis requires network service map information in addition to network topology information.

A network service is any program providing a service to remote machines over the network, and ranges from Link-layer services such as DHCP and AAA and Network-layer services such as DNS, to Application-layer services such as SMTP server, Web server, VoIP server, Terminal service, NFS server, directory server, DBMS server, etc. Given an enterprise or data center network, how to automatically discover all network services deployed on the network and which hosts have accessed which network services are the main research problems to be addressed in this project.

1.1 Active Service Discovery

Service discovery can be achieved by either *active service probing* or *passive network packet analysis*. Active service probing works by sending packets to a host with predefined format or content. The hosts responses will then be tracked and compared with its corresponding signatures. To efficiently discover the available services, host discovery will be performed first to check the presence of a host. After that, all ports on the target host will be scanned [3].

A well-known tool for active service probing is Nmap (Network Mapper) [7], which is an open-source security scanner designed to discover hosts and services in a network. It provides host discovery, port scanning, version detection and OS detection by sending the raw IP packets in novel ways to probe a network. For example, to detect services on a host, by default, Nmap first scans all ports of the host by sendingTCP SYN to that port, which is called ”TCP SYN scan”. If the port is listening, it responses with TCP SYN+ACK, otherwise it sends TCP RST (reset). By using this technique, Nmap can quickly know whether port is open or closed. To get further information about a port on a host, Nmap tries to determine the service and the version of the service by sending more probing packets. It establishes a connection to the ports and listens for five seconds. Many common services, including most FTP, SSH, MTP, Telnet, and POP3 servers, identify themselves in an initial welcome banner. If any data is received, Nmap compares it to a signature file and if matched, the service is fully identified. But what if the service does not have welcome banner? In this case, Nmap actively sends generic probes to the ports. For example, it sends two blank lines to the service and wait for any responses. If not matched, Nmap continues sending more probe packet, e.g. `"help\r\n"`.

The main drawback of active service probing is the large amount of traffic generated in the network. The port scanning technique can scan up to 65535 ports on one host and, furthermore, for each port, the number probing packets sent can range from one to around thirty, depending on the protocol signature. A blind scan without the any topology information will definitely cause problem. In brief, active probing requires an experienced administrator with a thorough scanning policy to detect network services without deteriorating overall performance.

1.2 Passive Service Discovery

Another method, passive network packet analysis, tries to infer the network services by (1) collecting the initial payload of every flow that traverses through an enterprise network, and (2) identify the type of network service based on the payload. Step (1) turns out to be quite difficult in practice for typical enterprise networks because common network flow summary formats, Ciscos NetFlow, IETFs IPFIX (used by Nortel), and Sflow (used by Juniper), support only per-flow header information (at various layers) but not per-flow payload information. Step (2) is also non-trivial because of the large number of protocols used in enterprise networks, some of which may even be proprietary.

1.2.1 Collecting flows through entire network

The simplest approach to recognizing the network service behind a network flow is to identify the service based on the flows source and destination port numbers. Because NetFlow records include source and destination port numbers used in each network flow, this approach is relatively straightforward to implement and requires minimal modification to the enterprise network. However, this approach has a major drawback: It cannot recognize all network services, especially those that do not have well-known port numbers or those that hide behind well-known port numbers (e.g. Port 80). Moreover, because collecting NetFlow records incur a substantial performance overhead, it is essential that routers turn on the NetFlow feature only when absolutely necessary. How to decide when to turn on and off the routers NetFlow feature so as to simultaneously minimize the performance overhead and maximize the network service discovery accuracy is a research subject of this project.

IPFIX is an IETF standard derived from NetFlow v9. It supports a self-describing mechanism by including in each IPFIX message a template part that describes the data being reported and a data part that contains the actual reported data, and is thus more extensible than NetFlow. IPFIX messages can be sent to a central collector through STCP, TCP or UDP. Although IPFIX is extensible, it is expected than the actual contents in IPFIX messages are largely the same as those in NetFlow messages, and therefore will not contain any per-flow payload information either.

sFlow is a sampling-based packet reporting technology that samples packets going through a network interface according to a time or packet count, which is configurable. Network devices typically report sampled packets to a central collector using UDP. sFlow is more efficient because it does not need to recognize flows and keep per-flow state. In addition, sFlow messages contain packet payload information, which could be used for more accurate network service recognition. However, the sampled packets that sFlow reports do not necessarily reflect all the flows traversing a network, making it possible to miss some network services during the discovery process. sFlow is similar in spirit to the remote port mirroring technology discussed below.

To go beyond port number-based network service identification, one needs to capture and analyze the payloads of network flows. To obtain a comprehensive view of the network flows traveling on an enterprise network, it is preferable to collect their payloads on intermediate network devices such as Layer-2 switches rather than on end hosts to reduce the total number of data collection devices required. The only known way to do this is port mirroring or port monitoring, which allows a switch to copy ingress or egress traffic through one or multiple switch ports (called source ports) to another switch port (called destination port). A more advanced version of port mirroring is remote port mirroring, which allows a switch to copy traffic not to a physical local switch port, but to a special virtual LAN (VLAN) to which one or multiple data collection machines are connected, and thus makes the number of data collection devices required largely independent of the size of the network being monitored. For example, given a 10-switch network, remote port mirroring makes it possible to use a single physical machine to collect all traffic going through all ports on these 10 switches. This feature is called RSPAN on Cisco switches.

1.2.2 Limitations

Recent works [1] and [2] have shown that RPSAN and SPAN do not *flawlessly* copy data from the monitoring port but instead have some effects on the mirroring packets. The effects are 1) change of the frame timing: the timing of a frame might be different because of the overhead introduced by SPAN/RSPAN. 2) frames are dropped: when choosing between switching and spanning, definitely switching has the higher priority because its the main function of a switch. So if replicating a frame becomes an issue (e.g. port is over loaded), the spanning frame will be drop. 3) no frame delivery guarantee: since frame might be dropped, the data collected from the centralized monitor point might be incomplete. The current SPAN/RSPAN does not provide information about loss of frame or details about SPAN/RSPAN performance. To mitigate these effects, in this report, a sampling method called weighted round robin is presented.

Similar remote port mirroring capability is also supported on Nortel and Juniper switches. Given this remote port mirroring technology, the specific technical challenges in packet payload collection are:

1. How to automate the process of configuring the switches in an enterprise network to best exploit their remote port mirroring capability, especially when the network uses more than one routers,

2. How to sample network flow payloads and schedule port mirroring for different parts of the monitored network to minimize the performance overhead associated with payload collection, and

3. How to determine the minimum number of data collection devices required given a networks topology and workload characteristics are two research challenges.

   We tries to answer these questions in the next section.

1.2.3 Identifying the type of services

Once the payloads of network flows are collected, we need to analyze them to identify their corresponding network services. Wireshark (previously known as Ethereal) [5] is an open-source network protocol analyzer that currently supports more than 700 link-layer, network-layer, transport-layer and application-layer protocols, and also comes with a plug-in architecture for developing additional analysis engines for non-supported protocols. When Wireshark receive a packet, it identifies the protocol by finding the right packet dissector using some conventions, e.g. port number. However, this method is unreliable when random ports are used for some protocols. Wireshark tried to solve this problem by introducing its *heuristic dissector* (HD), which is a mixture of port-mappings, signatures and protocol data of other packets approach. It also provides extensibility for users to add rules to particular protocol dissector. Unfortunately only a small portion of protocols in Wireshark implements the heuristic dissector. For example, an HTTP service run on ports other than 80 can be misjudged by Wireshark. This makes its service discovery mechanism inaccurate.

We may, therefore, reasonably conclude that the best solution is to have *active service probing with an intelligent scanning scheme, while using passive network packet analysis with RSPAN as an auxiliary.* In this report, we present a hybrid solution to leverage both the active and passive scanning techniques. Once network services are identified, because flow reports include information on the network devices that generate them, we can also infer the set of network devices that a flow goes through, and eventually the set of network devices through which a network service is accessed. In other words, given a network, the proposed system can automatically infer (1) the set of network services running on it, and (2) the set of network devices through which each discovered network service is accessed.

2 Design and Implementation

In the preceding section, we have discussed 1) a centralized traffic collection method based on RSPAN on Cisco switches and 2) the techniques used in network service discovery. 3) Layer 2 topology discoveries. In this chapter, we first focus on providing an efficient way to minimize the performance overhead caused by port mirroring for different parts of the monitored network associated with RSPAN. After that, we will present the design of hybrid solution which leverages both the active and passive scanning techniques. Finally, a preliminary Layer 2 topology discovery method is proposed.

2.1 Weighted Round-robin RSPAN

As mentioned in previous section, by RSPAN, traffic can be copied from any data port to a single unused port, which makes centralized traffic collection possible. However, it suffers from loss of frames and inaccurate frame timing. In practice, its impossible to SPAN/RSPAN all incoming and outgoing traffic on each port of all switches in an enterprise network. The proposed weighted round-robin sampling method tries to prevent over-loading the switch port by scheduling the spanning time according to the throughput of the port. Spanning time of a port is defined as the time interval when SPAN/RSPAN selects this port to be monitored. The basic idea of round-robin RSPAN is to minimize the loss of frames due to sampling. Thus, a port with higher throughput has longer spanning time while a port with lower throughput has shorter spanning time.

The proposed method needs to first query the target switch and retrieve the current throughput on each port. The command line interface (CLI) can be used via a remote session to access the device with password as authentication. Applications such as Telnet and SSH can be used to establish the connection. To automate the configuration process, a simple program can be created to login to the switch and send commands accordingly. The calculation of the spanning time can be determined by finding the ratio of throughput of all other port and setting spanning time interval of each port according to their ratio.

2.2 Network Service Discovery

The service discovery application can be divided into three phases as follow.

Step 1: Nmap initial scan. To gain the first service map of a network, Nmap first fully scan all the hosts in the network. This method requires IP-subnet to be known in advance. The scan result provides each host with a network service map (e.g., which services are running on the host), and for each service with its application version information. This initial service map is saved into a file and later loaded into the application for query. Moreover, the full scan also provides OS detection results, which indicates the IP and Mac address of switches in the subnet. This information helps the network topology discovery to locate the first switch and traverse to all other switches. Fig. 1 shows the initial scan concept.

Step 2: Traffic monitoring/sampling. Fig. 2 shows monitoring/sampling

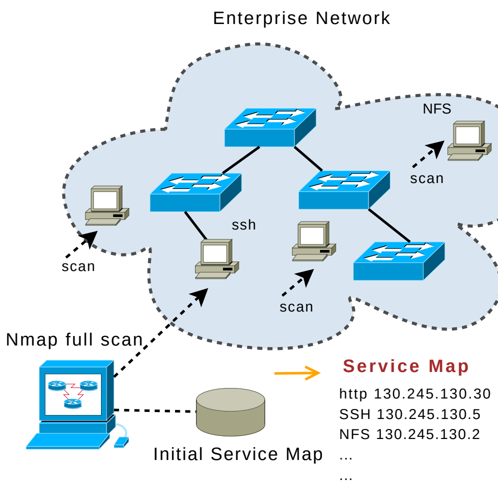

Figure 1: Nmap full scan. The initial service map is created.

state of service discovery application. After the initial full scan, the application enters monitoring and sampling mode, which passively listen to the traffic collected from RSPAN destination port. We first classify packets into flows, defined by the 5-tuple source IP, source port, destination IP, destination port, and protocol. A flow is further classified as identified if either the service run on source IP port or destination IP port has already been scanned by Nmap. In other case, a flow is tagged as unknown and saved in a future scan queue. A thread in the application will check the queue and, if non-empty, it invokes Nmap to scan that particular port on that host. This method achieves the discovery of new services by monitoring packets and avoiding the use of full scan.

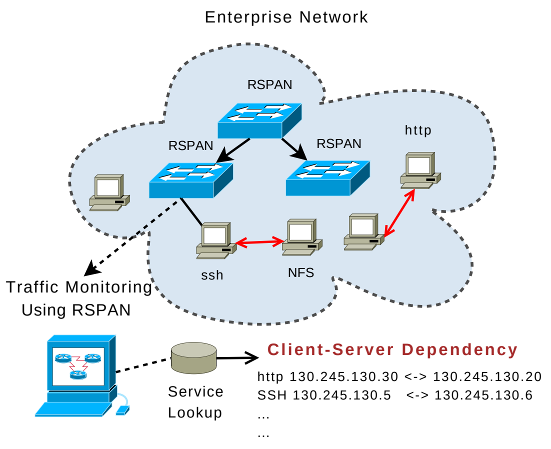
Enterprise Network
RSPAN
RSPAN
RSPAN
http
ssh
NFS
Traffic Monitoring Using RSPAN
Service Lookup
Client-Server Dependency
http 130.245.130.30 <-> 130.245.130.20
SSH 130.245.130.5 <-> 130.245.130.6
...
...

Figure 2: Traffic monitoring/sampling. Traffic data from other switch ports are copied to a centralizaion monitoring host via RSPAN

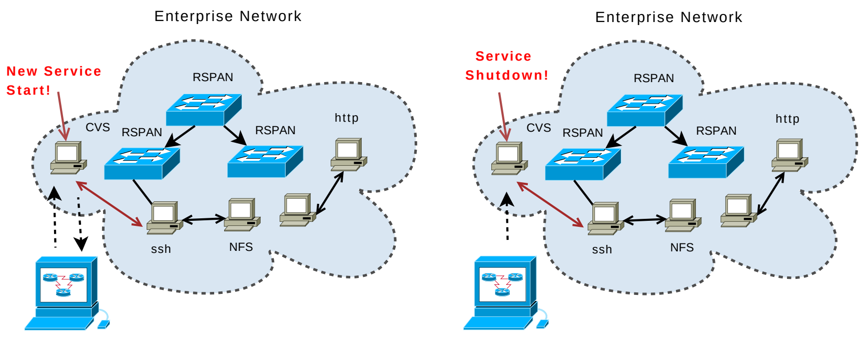

Figure 3: Updating the service map.

Step 3: Updating the service map. Due to the frequent changes of network services, the service map constructed from the first two phase may be obsolete. For example, an http server might be shutdown or migrated to another host some time after the initial full scan. Fig. 3 right shows that a server is shutdwon and the service provided by that host is no longer exists. Fig. 3 left shows a new service comes up. Both cases make the current service map incomplete. To solve this issue, a thread is periodically invoked and checked each service listed in the current service map by calling Nmap to scan it. If a service is no longer exists, we remove the entry in the service map. This makes sure that the service map represented is timely and accurate.

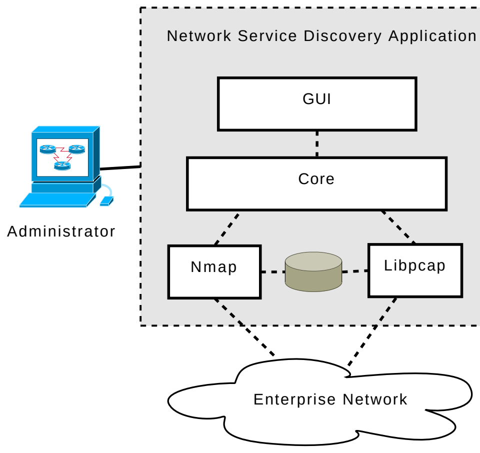

Figure 4: Function blocks overview of Service Discovery Application.

2.3 Implementation

This section describes an overview of function blocks in service discovery application we've implemented.

The function block in more detail:

- GUI: Java is selected as the GUI implementation language. The application layout will be discussed in the later section.
- Core: main glue code that controls GUI, Nmap and libpcap. This part is implemented in C language.
- Nmap: the command-line Nmap is integrated into the application. The scanning result is saved to a file as network service map.
- Libpcap: The packet capture library, including packet capture filter engine.

Experiment: we emulate a simple enterprise network in ECSL lab, which consists of one gateway, five switches and round 20 hosts running various applications. Network services in ECSL include http, https, ntp, nfs, http, ssh, cvsperver, Microsoft-ds, netbios-ssn, ipp, dns, ldap, telnet, rpcbind, and vpn. To get started, a vantage point is selected and the service discovery application is running on that host. The full scan time of the 20 hosts in ECSL can range from 15 to 20 minutes. This time varies depending on the link capacity and delay in the target network. Except vpn, all services are discovered by Nmap with correct port and version number. Every packet is analyzed by looking up the service map and dropped if the flow is already discovered. By manually starting an HTTP server and making a connection to it, the service discovery application can successfully detect the new flow and trigger Nmap to scan the host, then update the service map.

2.4 Link Layer topology

The topology discovery method relies on the address forwarding table (AFT) of switches. When a frame is received by a port, the source of the MAC address will be stored into the forwarding table. Using this forwarding table on each switch, the topology can be inferred by the Direct Connection Theorem [6]. So given the MAC addresses of all the ports in the switches and all the end hosts, and the forwarding tables on the switches, we can reconstruct the spanning tree underlying the switch network as follows:

Let's call the set of MAC addresses that switch I will forward using its port J as REACH[I,J]. There are three heuristics we can use to determine if port J of switch I is connected to port N of switch M.

1 If REACH[I,J] is a singleton, say H, that means switch I is connected to the end host H through port J.

2 If REACH[I,J] contains the MAC address of some ports in switch M, then switch M is reachable from switch I through port J.

3 If port J of switch I is directly connected to port N of switch M, then Intersection(REACH[I,J], REACH[M,N]) = empty and M is reachable from switch I through its port J and I is reachable from switch M through its port N.

In Rule 3, we assume every switch's forwarding table contains routing information for every MAC address (switch port or end host) in the switch network. This is not always the case, because forwarding table entries are typically expired after a while and need to be refreshed. Recent work in [4] tries to accomplish this problem by 1) generating additional ICMP traffic across switches to update entries in AFT, or 2) retriving the AFT information periodically to ensure the majority of the MAC addresses are collected. However, both methods do not garantee *complete* AFT. It concludes that a hybrid of both techniques with some reasonable approximation and heuristics work best in practice.

3 Planned Work

In the coming four months, we plan to 1) implement vendor-independent Layer 2 topology discovery algorithm and test it by using two Cisco Catalyst 2960 switches, 2) discover Layer 3 network topology by retriving the SNMP MIB variable (ipRouteTable) in each routerm, 3) integrate link layer and netwrok layer topology information. to provide end-to-end path for each host pair, 4) improve user interface, and 5) Test and refine the service discovery application in large networks.

References

[1] SPAN Port or TAP? CSO Beware(by Tim ONeill). *Available from World Wide Web: http://www.lovemytool.com/blog/2007/08/span-ports-or-t.html.*

[2] Using the Cisco Span Port for San Analysis. *Available from World Wide Web:http://www.cisco.com.*

[3] G. Bartlett, J. Heidemann, and C. Papadopoulos. Understanding passive and active service discovery. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 57–70. ACM New York, NY, USA, 2007.

[4] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz. Topology discovery in heterogeneous IP networks. In *IEEE INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 1, 2000.

[5] G. Combs et al. Wireshark-network protocol analyzer. *Available from World Wide Web: http://www.wireshark.org.*

[6] B. Lowekamp, D. O'Hallaron, and T. Gross. Topology discovery for large ethernet networks. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 237–248. ACM New York, NY, USA, 2001.

[7] G. Lyon. Nmap. *Available from World Wide Web: http://nmap.org/.*