A System for Logging Operation Histories of DLNA Devices by Combining ARP Spoofing and SSDP

Shunsuke Saruwatari
RCAST, The University of Tokyo

Johan Hjelm and Toshikane Oda
Ericsson Research Japan

Hiroyuki Morikawa
RCAST, The University of Tokyo

***Abstract*—This paper describes the design, implementation, and evaluation of DLNA Probe, a system for logging the operation histories of Digital Living Network Alliance (DLNA) devices. This paper first describes the combined use of Address Resolution Protocol (ARP) spoofing and the Simple Service Discovery Protocol (SSDP), which obtains the operation histories of DLNA devices without changing any DLNA specifications. Second, it shows that a unicast ARP request is the best of four kinds of ARP spoofing for DLNA Probe. Third, it shows the performance of the implementation of DLNA Probe.**

I. Introduction

The user activity and context data in home environments can provide useful services, and so researchers are studying methods to obtain or utilize the user activity and context. For example, the manufacturers of home devices are standardizing ECHONET and OSGi that can be used for obtaining the user activity and context. However, current home networks have few ECHONET and OSGi products.

As the first step to retrieve user activity and context from the home, we focus on DLNA-ready devices, which are currently coming into widespread use. The operation histories of DLNA devices can be converted into user activity and context when a user boots up a DLNA device or makes use of the content.

II. Operation Histories of DLNA Devices

We can use the operation histories of DLNA devices for context-aware applications, ethnography, recommendation services, and so on. For example, we can construct an automatic power control service by combining the operation histories and machine learning. The power control service saves power consumption and minimizes the user waiting time by automatically turning on/off each DLNA device according to the learned user activity [1].

Generic Event Notification Architecture (GENA) is another candidate to obtain the operation histories of DLNA devices. GENA, which is implemented in Universal Plug and Play (UPnP) networking, is able to give notification of the operation events of a DLNA device to synchronize multiple Digital Media Controllers (DMCs). However, if using GENA for the obtainment, we need to change DLNA specifications since GENA in the current DLNA can send notification of only some events, such as a play event in a Digital Media Renderer (DMR).

III. DLNA Probe

To obtain the operation histories of DLNA devices, we designed DLNA Probe, which monitors the communication among the DLNA devices that are already in real home environments. DLNA Probe does not need to change any DLNA specifications.

Figure 1 shows an overview of DLNA Probe. DLNA Probe is constructed with five components: a directory component, a discovery component, a spoofer component, a redirect component, and a logger component.

Directory Component: The directory component manages the information of DLNA devices as a directory service. The directory component receives a tuple from the discovery component. The tuple includes an IP address, a MAC address, and the discovery time. The directory component provides tuples as a device table for the spoofer component, the redirect component, and the logger component.

Discovery Component: The discovery component is constructed with an active search and a passive search. The active search discovers DLNA devices by periodically sending Simple Service Discovery Protocol (SSDP) packets, and notifies the directory component about the discovered devices. Although SSDP can actively discover DLNA devices with M-SEARCH, some DLNA clients do not reply to M-SEARCH. For example, some DMCs do not reply to M-SEARCH. To discover such DLNA clients, the passive search monitors M-SEARCH and NOTIFY, which are periodically sent by the clients. The discovery component obtains an IP address and a MAC address from messages with a raw socket as a PF_PACKET, and the messages include M-SEARCH, NOTIFY, and a reply to M-SEARCH.

Spoofer Component: The spoofer component rewrites ARP tables on DLNA devices that have IP addresses and MAC addresses in the device table, which the spoofer component obtains from the directory component. The rewriting of ARP tables is called ARP spoofing.

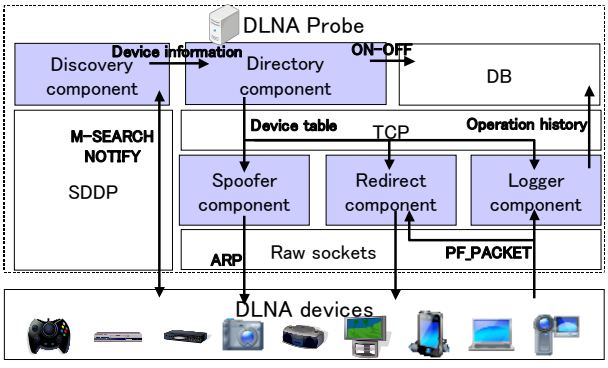

Fig. 1. Overview

TABLE I
COMPARISON OF THE FOUR METHODS OF ARP SPOOFING

	broadcast ARP reply	broadcast ARP request	unicast ARP reply	unicast ARP request
difficulty to be detected	×	×	×	○
rewrite area	all	all	a selected terminal	a selected terminal
traffic from DLNA Probe	△	△	○	×

Figure 2 shows an example of ARP spoofing. In this figure, DLNA Probe rewrites the ARP table on Terminal 1 (T1) and changes the direction of packets from Terminal 2 (T2) to DLNA Probe. Initially, the ARP table on T1 associates the IP address 192.168.0.2 with the MAC address 22:22:22:22:22:22. 1) In this situation, when T1 sends a packet to 192.168.0.2, T2 receives the packet. 2) DLNA Probe sends a forged ARP packet to T1. 3) The forged packet rewrites the MAC address from 22:22:22:22:22:22 to 33:33:33:33:33:33. 4) After the rewrite, when T1 sends a packet to 192.168.0.2, DLNA Probe receives the packet.

As shown in Table I, there are four kinds of ARP spoofing, depending on whether ARP uses a unicast or a broadcast and whether it uses an ARP request or an ARP reply. DLNA Probe uses the unicast ARP request for two reasons: ARP spoofing with the unicast ARP request is not detected by DLNA devices, and only DLNA devices can be spoofed. The unicast ARP request rewrites the ARP table on only a destination terminal. Therefore, a hijacked terminal and a spoofed terminal cannot detect the ARP spoofing.

Redirect Component: All packets of DLNA devices are sent to DLNA Probe because of the spoofer component. The redirect component redirects the received packets to the correct destination by rewriting the MAC addresses in the packets.

Logger Component: The logger component examines the payload of packets redirected by the redirect component. Since the purpose of DLNA Probe is to obtain the operation histories, DLNA Probe stores the HTTP headers because all the operations of DLNA devices use HTTP.

IV. EVALUATION

To evaluate the basic performance of DLNA Probe, we implemented it on Linux 2.6.27-17, and evaluated its delay and throughput.

First, we measured the average delay among DLNA devices. When we use DLNA Probe, the delay among DLNA devices must be less than 7 ms to meet the requirements of Digital Transmission Content Protection over Internet Protocol (DTCP-IP), which protects copywrited contents. As shown in Figure 3, when we use DLNA Probe, the average delay increases to approximately 1 ms. When the number of devices is 100, the average delay is approximately 1.6 ms.

Second, we measured the throughput among DLNA devices. It is a problem if DLNA Probe seriously decreases the throughput because DLNA includes video streaming services that need several tens of Mbps of throughput. To evaluate the effect of DLNA Probe on the throughput, we measured the throughput on two flows with netperf and iperf on four terminals, labeled A, B, C, and D. We used netperf for measuring the maximum TCP throughput from A to B, and we used iperf for generating static UDP traffic from C to D.

The measurement results are shown in Figure 4. The horizontal axis represents the setup bandwidth of iperf from C to D. The vertical axis represents the measured throughputs. In all iperf settings, the sum of the throughput of A to B and C to D is not more than 94 Mbps because all packets pass through DLNA Probe and the total throughput is limited by the link speed on DLNA Probe.

V. CONCLUSION

This paper presented DLNA Probe, a system for logging operation histories of DLNA devices by combining ARP spoofing and SSDP. Currently, we are gathering operation histories in real home environments.

REFERENCES

[1] H. Si, et al. A ubiquitous power management system to balance energy saving and response time based on device-level usage prediction. *Journal of Information Processing*, 18:147–163, 2010.

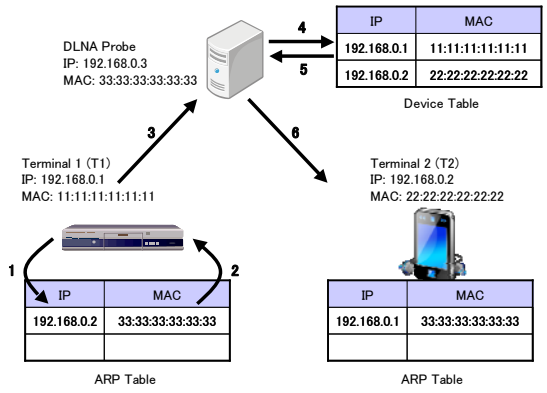

Fig. 2. ARP spoofing

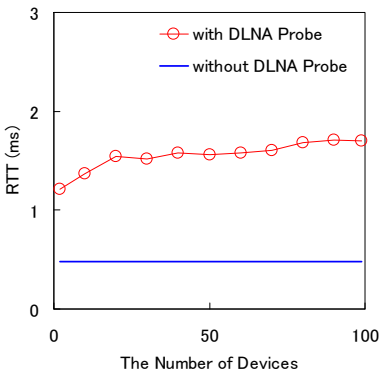

Fig. 3. Average delay vs. number of devices

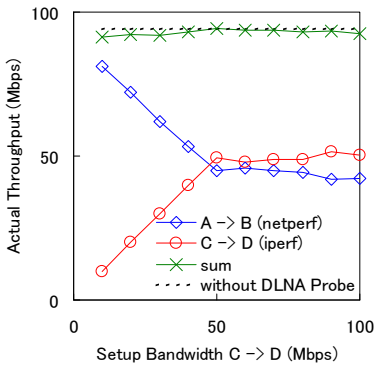

Fig. 4. Throughput of two flows