Host Identity Detection in IPv6 Networks

Libor Polčák(✉), Martin Holkovič, and Petr Matoušek

Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
{ipolcak,matousp}@fit.vutbr.cz, xholko00@stud.fit.vutbr.cz

Abstract. It is important to keep networks secure and reliable. In order to backtrack security incidents, provide accounting for offered services etc., it is necessary to know the identity of network users. With various methods for IPv6 address assignments, user identification in IPv6 networks is challenging. This paper proposes a new approach for user identity tracking in LANs. The approach is based on network control traffic that is already present in IPv6 networks. In contrast to current methods, the proposed approach does not bring any extensive workload to active network devices and works in networks with Multicast Listener Discovery snooping. In addition, the approach is able to detect that an address is no longer used. The proposed approach is passive to end devices. In order to make the approach reliable, we studied the behaviour of current operating systems during IPv6 address assignments. We implemented a tool called *ndtrack* based on the proposed approach and tested it in a real network.

Keywords: Computer network security · Host identity · IPv6 monitoring · SLAAC · Neighbor Discovery

1 Introduction

In order to maintain a reliable network, its administrators need to monitor the network and its weak points, detect misuse of the network, backtrack security incidents, provide accountings for the offered services etc. The knowledge of the identity of computers and their users in the managed network is essential to achieve these tasks. The imminent exhaustion of IPv4 address space and the transition to IPv6 [1] requires keeping track of IPv6 addresses used in the managed network.

The IPv6 architecture guarantees at least 2^{64} addresses allocated to each LAN [2]. There are several mechanisms that manage the allocation of addresses. For example, Stateless Address Autoconfiguration (SLAAC) [3] allows an end device to generate as many IPv6 addresses as it needs, e.g. for privacy concerns [4,5], as long as the addresses are not already used by another device in the network. Note that the addresses are not handled centrally but generated by end devices. Moreover, the network operator is not able to influence the address

M.S. Obaidat and J. Filipe (Eds.): ICETE 2013, CCIS 456, pp. 74–89, 2014.
DOI: 10.1007/978-3-662-44788-8_5

generation process. In addition, there is not a node in the network that keeps track of IPv6 addresses being used by devices connected to the network. Even more, a host does not send any specific message when an address is no longer used by the host.

One possibility of user identity tracking is through authentication. For example, RADIUS authenticates a user and MAC address of his or her device connected to a network. In some networks, such as the campus network at our university, users have to register their MAC address before they are allowed to access the Internet and services offered in the network. However, unlike network layer addresses, MAC addresses are not propagated outside of LANs. Hence, the knowledge of bindings between network layer addresses and MAC addresses is crucial for network management, security, and accounting.

In this paper, we propose a new approach for the problem of the user identification in IPv6 networks accessed by users with devices that are not under direct control of the network operators, i.e. the problem of learning the MAC and IPv6 address pairing. In order to achieve this, we studied the implementation of IPv6 in current operating systems (OSes). As a result, the approach is suitable for campus-wide networks, networks of companies that allow their staff or customers to connect their own devices to the network ("bring your own device" policy), Wi-Fi and Ethernet hot spots, and hotel networks.

One of the contributions of this paper is the identification of the differences in the behaviour of current OSes and their violations of RFCs concerning IPv6 address assignments. The main contribution of this paper is the proposal of a new approach for user identification in IPv6 networks, which is based on monitoring control messages that are already present in the network.

The proposed approach does not influence or modify IPv6 in any way. Since the method is completely transparent for network hosts, it does not require any additional changes in the network hardware or software. Moreover, the privacy of the users in the network, with respect to the outside world, is not affected by the proposed mechanism. The approach was successfully tested in a real network. In contrast to previously published methods, the proposed approach does not need any significant additional load on network devices, it can detect the exact time of address assignment even if the address is not used for communication, and the approach detects that an IPv6 address is no longer used.

This paper is organized as follows. Section 2 overviews the address assignment mechanisms in IPv6. The results of the study of the behaviour of current OSes during IPv6 address assignments are presented in Sect. 3. Section 4 outlines the proposed approach for user identity tracking in LANs. Section 5 discusses the related work and discuss alternative approaches. Our experiments are summarised in Sect. 6. Section 7 concludes the paper.

2 Preliminaries

This section reviews the basics of *Duplicate address detection* (DAD), a part of *Neighbor Discovery* (ND), and overviews common methods for IPv6 address assignments.

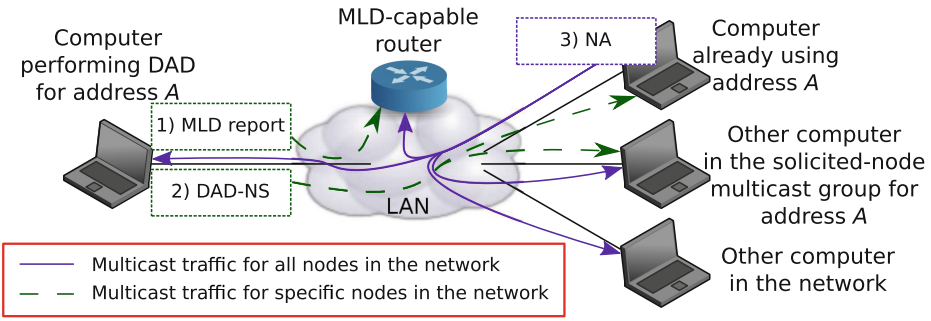

Fig. 1. Messages exchanged during DAD. As another computer is already using address *A*, the computer performing DAD cannot use the address.

2.1 Duplicate Address Detection

When a new IPv6 address is about to be used by a device, the device needs to test that the address is not already used in the network [3,6]. Until the new address is proven to be unique, it is called *tentative*. In order to prove that the tentative address is unique, the device has to perform the following steps (depicted in Fig. 1):

1. The device joins the solicited-node multicast group [2] whose address is derived from the tentative IPv6 address using *Multicast Listener Discovery* (MLD) protocol [7]. The request (MLD report) is sent to a multicast group of all MLD-capable routers – ff02::16.
2. The device issues a *Neighbor Solicitation* (NS) request to the solicited-node multicast group whose address is derived from the tentative address. In this paper, NS requests issued during DAD are denoted as *DAD-NS*.
3. If the tentative address is already used by another device, the other device should reply with a *Neighbor Advertisement* (NA) to the multicast group for all nodes in the network (ff02::1). Only if no NA is received before a timeout, the new address can be used.

To avoid race conditions in address assignments, RFC 4862 requires [3] that each host has to join the solicited-node multicast group before it sends the DAD-NS, i.e. step (1) has to be performed before step (2). Note that requests sent during step (1) and (2) are delivered only to the network hosts that are members of the respective multicast groups in network with enabled *MLD snooping* (i.e. multicast does not behave as broadcast).

As a result, there is no central point in the network that gathers all active addresses; the knowledge is spread over the network and is available through the solicited-node multicast groups.

2.2 Multicast Group Management

In general, in order to detect empty multicast groups, a dedicated router (or a group of routers) may be configured as an *MLD querier*. The MLD querier

periodically queries multicast groups to verify that some hosts are still part of each multicast group. If there is at least one host in a multicast group, the MLD querier receives a reply that the multicast group is still active in the network.

MLD querying is also performed for solicited-node multicast groups. If no address corresponding to the queried solicited-node multicast address is used, there is no reply in the multicast group. However, if more than one IPv6 address in the network coincide into one solicited-node multicast group, only one of the hosts replies. Such coincidence are rare due to the addressing scheme used for solicited-node multicast groups, which was specifically designed to reduce such coincidences.

2.3 Methods for Address Assignment

SLAAC [3] is a basic method for address assignments in IPv6. In contrast to DHCP, dominant in IPv4, SLAAC is not based on leases. Instead, a device itself generates its addresses. First, the device learns the network (higher) part of IPv6 address from a *Router Advertisement* (RA), a message periodically send by gateways in the network. Then, the device generates the lower part of the IPv6 address called *interface identifier* (IID) [2]. The original method for selecting an IID uses modified EUI-64 IID [2]. Later, privacy extensions [5] introduced completely random IIDs which may change during time. Besides privacy extensions, Windows machines use random IIDs [8,9]. In all cases, the uniqueness of the selected IPv6 address has to be proven by DAD.

Although there is a variant of DHCP called stateful DHCPv6 [10], it does not provide the same information as DHCP since DHCPv6 assigns IPv6 address according to *DHCP Unique Identifier* (DUID). DUID is generated by each host, mostly during OS installation. As a consequence, DUID is changed when a host is rebooted to another OS. Therefore, the MAC and IP address pairings are not stored in DHCPv6 logs. On the other hand, the assigned address has to be confirmed by DAD.

Finally, it is possible to assign a static address. Whenever a new static IPv6 address is entered on a host, it has to be validated by DAD.

As mentioned above, DAD, or more generally ND, is a part of each mechanism for address assignments. Hence, we choose ND as a basis for the proposed network monitoring approach. However, as discussed in Sect. 3, ND is not implemented in the same way among current OSes. Based on this study, Sect. 4 proposes a new method for monitoring IPv6 addresses in a network without knowledge of OSes installed on the hosts in the network.

3 Study of Operating System Behaviour During ND

This section describes implementation of ND in current OSes. We studied [11] the exact sequences of messages that are issued during DAD after an address is assigned or automatically generated. The main goal was to validate that OSes follow the sequence ordered by RFC 4862 [3] (discussed in Sect. 2). However, the results are not positive and some OSes diverge.

Table 1. Operating systems tested for compliance with RFCs specifying ND.

OS family	Tested variants
Windows	XP SP3, Vista, Vista SP3, Server 2008 R2, 7, 7 SP1, and 8
Linux	various distributions including Red Hat, CentOS, Debian, and Ubuntu (kernels 2.4.27–3.2)
Mac OS X	10.6.2 (kernel 10.2)
Unix	FreeBSD 9.0, OpenBSD 5.0, and Solaris 5.11

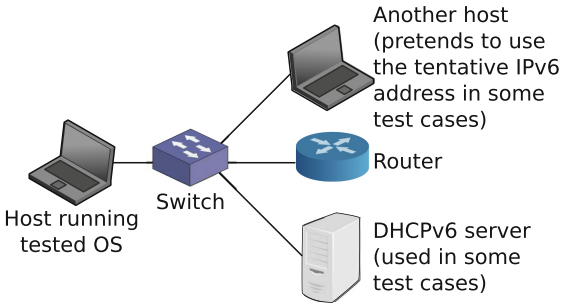

Fig. 2. The network topology used for OS behaviour study. DHCPv6 server and the another host were active only in specific test cases.

Firstly, we selected OSes (see Table 1) that we believe are the most common in current LANs. Then, we connected hosts running these OSes into our laboratory network (see Fig. 2) and captured all packets that were sent or received by each host in all test cases.

Several experiments (test cases) were performed during the study:

- In test cases focused on SLAAC, the router in the network (see Fig. 2) sent an RA. Consequently, the tested host generated link-local IPv6 address and one or more IPv6 address from the scope advertised by the router. We analyzed the messages issued by the tested host during DAD for the generated addresses.
- Next set of test cases focused on DHCPv6. DHCPv6 server in the network (see Fig. 2) was activated and the tested hosts were configured to use it. One of the goal of this test was to confirm that OSes perform DAD for IPv6 addresses obtained from a DHCPv6 server.
- Next set of tests aimed at DAD during static address assignments.
- In order to test the behaviour of hosts in the presence of different hosts having the same IPv6 address, a different host was connected to the network (see Fig. 2). A specialized program that is able to spoof NAs was running on the added host. Consequently, the computer was able to simulate a collision of all IPv6 addresses; including randomly generated RFC 4941 addresses. The goal was to validate that the examined host does not use any address that is claimed to be used in the network.

The evaluation of the packet traces [11] gathered from the test cases revealed following anomalies:

1. Windows Vista and later (for all IPv6 addresses) and FreeBSD (for static and EUI-64 IPv6 addresses) ignore DAD-NS if the other host has the same MAC address.
2. Windows Vista and later use a tentative link local address to join multicast groups before they start DAD for the address.
3. Solaris (for all addresses), FreeBSD (for static addresses), and Windows Server 2008 R2, Windows 7 and later (for link local addresses) diverge from the recommended sequence of actions during DAD because they send DAD-NS before they join the solicited-node multicast group derived from the tentative address.
4. Windows 8, Solaris, and Mac OS X send NAs during DAD that are not mandatory. These NAs are destined to the all-nodes-in-the-network multicast group (ff02::1).
5. OpenBSD does not join solicited-node multicast groups at all.
6. All selected OSes that register to multicast groups reply to MLD queries. However, we discovered that some OSes do not meet the maximal timeout specified in an RA. Although the RA announced maximal timeout of 0.1 s, some acknowledgements were received after 0.7–0.8 s.

The results of the study show that there are various differences in behaviour of current operating systems. The behaviour of FreeBSD is even not consistent and depends on the type of the IPv6 address that was generated.

4 Proposed Approach for Address Assignment Detection

The study of the behaviour of current operating systems presented in Sect. 3 allowed us to propose a new approach for address assignment detection in IPv6 networks [12]. The life-cycle of an IPv6 address A can be tracked by the extended Mealy FSM depicted in Fig. 3. Network control traffic is used as the input of the FSM, the output is the information that the address started or ceased to be used. The FSM is extended with a variable to store the MAC address of the interface that uses the tracked IPv6 address.

4.1 Inputs of the FSM

The FSM tracks ND messages to detect address assignments. MLD queries and responses are used to detect that the address is no longer used. Hence, we recommend to enable MLD querying on a router in the network.

In addition, if MLD snooping is active in the network (multicast traffic is not broadcasted), it is necessary to provide DAD-NSes and NA replies to the network host running the proposed FSM.

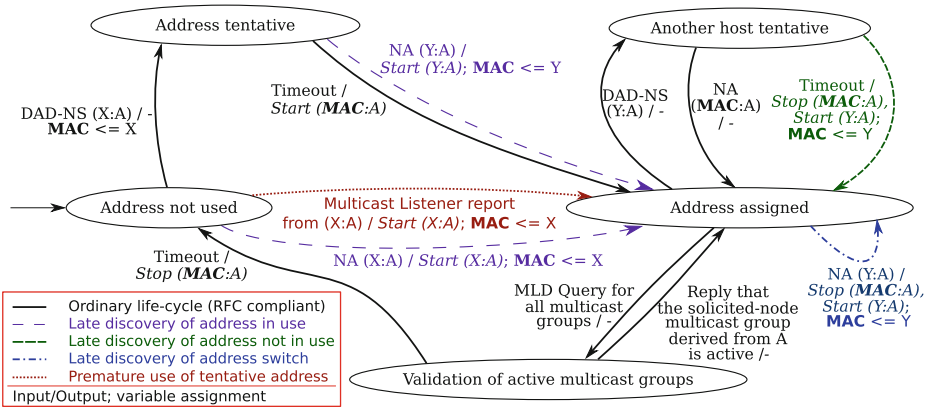

Fig. 3. The life-cycle of an IPv6 address A can be monitored by the Mealy FSM extended by the variable **MAC** to store the MAC address of the interface that currently uses A. Binding between MAC address X and A is denoted as (X:A).

– In case that OpenFlow switches [13] are available in the monitored network, we recommend simulating the FSM on the OpenFlow collector(s) and configuring OpenFlow switches to redirect (or copy) all DAD-NSes, NAs, and MLD queries to the collector(s);
– otherwise, the computer simulating the FSM should
  1. join the multicast group for *all nodes* (ff02::1) and *all MLDv2-capable routers* (ff02::16),
  2. detect all requests to join solicited-node multicast groups,
  3. join the detected groups.

Additionally, if a frame with source or destination address A is seen on the collector in OpenFlow-based networks, the FSM can treat them in the same way as if NA for given MAC/IPv6 address pair was received.

4.2 States and Shifting Logic of the FSM

The initial state of the FSM is the *Address not used*. When a host generates the address A, it issues DAD-NS, and the FSM shifts to *Address tentative*.

If everything worked according to RFCs, the only trigger for transition from *Address tentative* would be the timeout as the address would not be used in the network. However, Solaris, some versions of Windows, and FreeBSD do not join the solicited-node multicast groups before issuing the DAD-NS. As a consequence, in networks with MLD snooping, some address assignments may have been unnoticed earlier and the address can already be used. Therefore, it is possible to receive an NA from another host. In both cases the FSM detects the MAC address that is bound to the IPv6 address and shifts to *Address assigned*.

Similarly to the NA received after DAD-NS, the FSM may detect an NA for the address A in the initial state (e.g. non-mandatory NA during DAD).

Consequently, the FSM shifts directly to *Address assigned*. Additionally, the FSM shifts between these two states in case of Windows using a tentative link local address as the source address to join multicast groups.

In order to detect that the address was dropped by the host, *Validation of active multicast groups* is reached after an MLD query is received. In case that the solicited-node multicast group derived from address A is acknowledged, the address is most likely being used as the solicited-node multicast groups were designed so that two hosts are not likely to be in one solicited-node multicast group. If the MLD query expires, the address is definitely not used any more, and the FSM returns to the initial state.

While the FSM is in *Address assigned*, another host might try to use the address. When DAD-NS is received, the FSM shifts to *Another host tentative*. Ordinarily, the address is still in use and the NA follows. If the address was no longer used but it had not been detected (e.g. because the MLD query was not issued, yet), the NA would not be sent. In such case, the FSM also shifts back to *Address assigned*, however, the detected MAC address changes. In a rare occasion when an NA from another MAC address is seen in *Address assigned*, the FSM loops in this state and the MAC addresses are swapped. This loop is present in the FSM only for safety reason as the study of OS behaviour does not suggest that it is needed.

5 Alternative Approaches

This section discusses related work and alternative approaches for the detection of pairings between IPv6 addresses and MAC addresses. Later, these methods are compared to the detection approach proposed in Sect. 4 and advantages and disadvantages are identified.

5.1 Related Work

Several attempts have been made to study IPv6 IIDs. Groat et al. [4] proposed to use *ping* or *traceroute* to monitor a location of a node that uses a static interface identifier of any sort. Dunlop et al. [9] list an example of Windows using random, yet static networks IIDs. However, both papers aimed at global tracking of a movement of a specific user. In contrast, we want to monitor only the local network. In addition, both Groat et al. and Dunlop et al. need to know the address in advance. Our research is concerned with unknown IPv6 addresses. Another difference is that we want to learn all addresses that belong to every device connected to a network. Moreover, the proposed approach is passive to end devices.

Similarly to our goal, Grégr et al. [14] are also interested in learning the IPv6 addresses that were used by a host with a specific MAC address. They presented a campus network monitoring system which gathers data from the *neighbor cache* (NC) of the routers in the network using SNMP. However, two conflicting requirements needs to be balanced. In order to have sound information about

the IPv6 addresses in the network, they need to poll routers sufficiently often. Since routers are critical devices and the polling results in additional workload, the polling cannot be too frequent. As a consequence, a new address selected by a device in the network is learned with a delay. During a security incident, an attacker can use an address for a limited time. Consequently, the monitoring system can miss that the IPv6 address was used in case the expiration timeout of NC records is shorter than the polling interval.

Groat et al. [15] studied the possibility of using DHCPv6 for monitoring the identity of users in the network. The proposed approach is more general as it is not restricted to DHCPv6.

A tool called *addrwatch* [16] can monitor ND messages in the network. However, we identified several weaknesses of the tool. Firstly, *addrwatch* just reports ND messages. They are not put in any context. Secondly, *addrwatch* completely ignores multicast messages, therefore, it cannot detect address assignments in a network with MLD snooping (multicast is not broadcasted). Finally, *addrwatch* does not detect that an address is no longer used.

Asati and Wing [17] deal with the same problem as we do. However, their solution involves changes in routers behaviour. The proposed approach does not need any change on any critical network device.

5.2 Discussion

Although we have not found a reference, we expect that some administrators use port mirroring and parse the mirrored traffic with a sniffer to learn the MAC and IP address bindings. The proposed approach does not depend on processing all network traffic and consequently it is more efficient. In addition, the bandwidth of the mirroring port could be insufficient for all traffic traversing the mirroring switch or even one full-duplex port. Moreover, in case of mirroring the port connected to the router in the network, the learned MAC and IP address pairings are more or less the same to these stored in the NC of the router.

The amount of traffic for processing can be reduced in networks with OpenFlow switches [13]. In addition to ordinary traffic processing, OpenFlow switches send a copy of the first packet of each new flow to a central point in the network – a controller. The controller is a programmable device and therefore can be instructed to collect information about detected usage of IP addresses by devices connected to the network. However, this simple approach only reveals the address if it is actively used. To track the correct time of address assignments and address drops, we recommend following the instructions in Sect. 4.

Table 2 summarizes the identified methods for detection of IPv6 assignments. Two methods gather information from NC. The method proposed by Asati and Wing [17] failed in IETF standardization process and as it requires changes in the router behaviour, it cannot be used in practice. Neighbor cache polling (NCP) detects both address assignments and address drops with a delay. Detection delay depends on the frequency of polling while the detection of inactivity depends on the NC record expiration time. As NC is not available on switches, both methods cannot detect addresses used for internal communication.

Table 2. Methods for monitoring of usage of IP addresses.

Source of informations	Methods
Neighbor cache	Neighbor cache polling (NCP) [14], Asati and Wing [17]
Network control traffic	Proposed approach, *addrwatch* [16]
Network traffic	Parsing of mirrored traffic, Simple OpenFlow-based monitoring

The proposed approach is based on very similar principles to addrwatch. The main advantage of the proposed approach is that the FSM correlates the detected messages. In addition, the proposed approach joins the multicast groups (in networks without OpenFlow switches) and therefore works in networks with MLD snooping, except OpenBSD and static addresses in FreeBSD (see Sect. 6 for more details). As the detection of address assignments in networks with MLD snooping depends on joining solicited-node multicast groups, the approach may fail in larger networks where delays prevent the monitoring device from joining the solicited-node multicast group in time. Nevertheless, the proposed approach never yields worse information than *addrwatch*. In combination with OpenFlow switches, the proposed approach detects all address assignments, even in networks with MLD snooping (see Sect. 6).

As the parsing of mirrored traffic results more or less in similar information to the information gathered by NCP, it is not practical. Simple monitoring using OpenFlow (i.e. without information from control traffic) has several advantages in comparison to NCP: (1) the additional traffic load on the data plane is negligible as OpenFlow switches are optimised for copying traffic and passing it to the controller, (2) the controller detects active addresses immediately, and (3) it is possible to detect that an IP address is no longer used for active communication by setting OpenFlow timeouts for active flows. Nevertheless, OpenFlow monitoring combined with the proposed approach detects the exact time of address assignments and it can detect if an address was already dropped or if it is not used for active communication and thus, the address can be used for communication in the future.

6 Experiments

This section describes the experiments with *ndtrack* [18], a tool that follows the proposed approach for IPv6 address assignments that is sketched in Sect. 4 (specifically the approach tracking multicast groups).

The first experiment aimed at real network monitoring. As most of the devices were not under our control, we could not confirm that *ndtrack* detected all devices. However, we checked that all our devices were detected. The other three experiments were performed in a laboratory network; they focused on the quality of the monitoring approach.

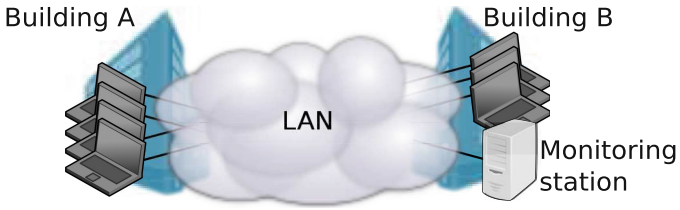

Fig. 4. The network topology of the real network monitoring. Some of the computers were not under our control.

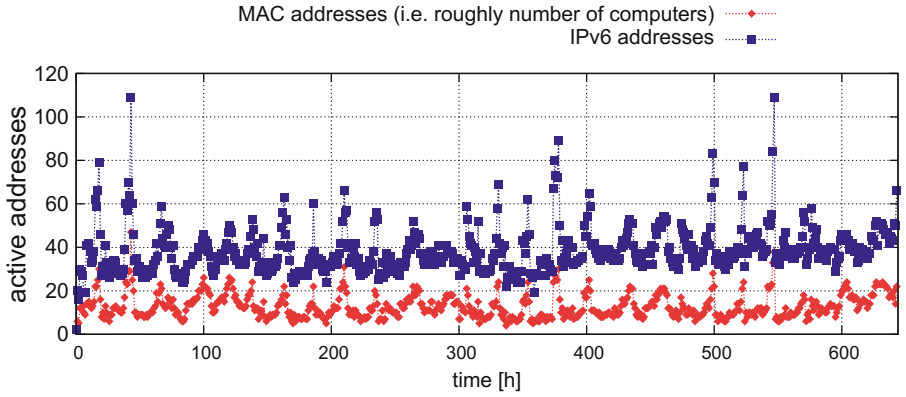

Fig. 5. Statistics of the real network monitoring. IPv6 addresses were successfully detected to be used (the number of known addresses rises during working hours) and dropped (the number of IPv6 addresses lowers at night).

6.1 Real Network Deployment

The first experiment aimed at long-term monitoring (almost a month) of SLAAC in a network with MLD querying enabled. The network spans two buildings and is available for all employees of the faculty (see Fig. 4).

We successfully validated that *ndtrack* detected IPv6 addresses of devices under our control among other devices of our colleagues that were being used in the network. Then we validated that the addresses are correctly identified as no longer assigned after the hosts disconnect or stop using the addresses (see Fig. 5 for the statistics).

In order to make the experiment more convincing, we connected a device to the network in a different building than the one in which the monitoring station was located. All addresses assigned to the device were correctly identified and later dropped when we disconnected the device.

6.2 Network with MLD Snooping

We validated the behaviour of *ndtrack* in the presence of MLD snooping in our laboratory. A monitoring station running *ndtrack* and a testing computer were

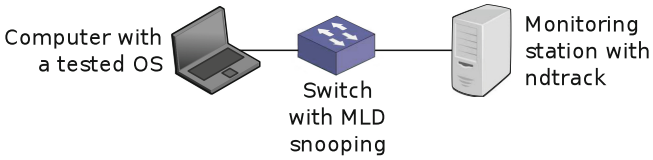

Fig. 6. The network topology for the experiments in a network with active MLD snooping.

Table 3. Effectivity of the proposed approach (✓ = Detected) in networks with active MLD snooping — without (⊖) and with (⊕) joining the solicited-node multicast group. In addition, the table shows expected effectiveness in OpenFlow-based networks (⊙).

OS	Static addresses			SLAAC addresses		
	⊖	⊕	⊙	⊖	⊕	⊙
Windows 7 and earlier	-	✓	✓	-	✓	✓
Windows 8	✓	✓	✓	✓	✓	✓
Linux	-	✓	✓	-	✓	✓
Mac OS X	✓	✓	✓	✓	✓	✓
FreeBSD	-	-	✓	-	✓	✓
OpenBSD	-	-	✓	-	-	✓
Solaris	✓	✓	✓	✓	✓	✓

connected to a switch with MLD snooping enabled as depicted in Fig. 6. In the first set of experiments, *ndtrack* did not follow the advice given in Sect. 4 and did not join the appropriate multicast groups. In the second set of experiments *ndtrack* joined the multicast groups as recommended in Sect. 4. Several OSes were tested during each set of experiments.

The results of the experiment are summarised in Table 3. When *ndtrack* did not join the multicast groups, DAD-NS were not propagated by the switch in the network. Consequently, *ndtrack* was not able to learn the identity of computers that follows the recommended sequence of messages during DAD. Windows 8, Mac OS X, and Solaris send additional NAs (as discussed in Sect. 3) which allowed *ndtrack* to learn their identity without joining the multicast groups. When *ndtrack* was a member of the specified multicast groups during DAD performed by the tested computer, *ndtrack* successfully detected all OSes except OpenBSD and static addresses in FreeBSD.

OpenBSD does not join the solicited-node multicast groups derived from the tentative address. Consequently, *ndtrack* did not know that it should join the solicited-node multicast group. As a result, the switch with activated MLD snooping did not propagate the DAD-NS to the monitoring station.

As already stated in Sect. 3, FreeBSD sends DAD-NS for a static address before it joins the solicited-node multicast groups derived from the tentative address. Therefore, *ndtrack* joined the solicited-node multicast groups derived

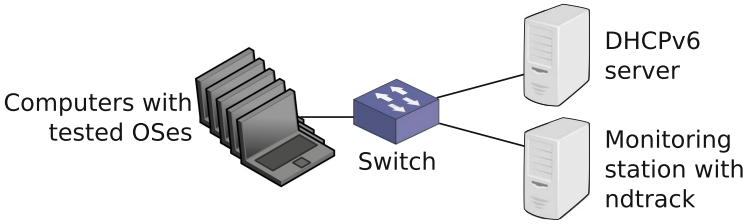

Fig. 7. The network topology for the experiments in a network with stateful DHCPv6.

from the tentative address after the DAD-NS was sent and as a consequence did not learn about the address assignment. Windows and Solaris that also join the solicited-node multicast group late (see Sect. 3) send additional NAs during DAD and therefore, were detected by *ndtrack*.

The OpenFlow-based proposed approach would have delivered all NAs to the OpenFlow controller and consequently to the monitoring center and therefore, all address assignments can be detected (see Table 3).

6.3 Network with Stateful DHCPv6

The next experiment tested stateful DHCPv6. We connected computers running Windows 7, 8, 2008 R2, Ubuntu 12.10, and Solaris (one computer for each OS) to a laboratory network depicted in Fig. 7. We verified that *ndtrack* detected all address assignments. Hence, DHCPv6 leases are detected by the proposed approach.

6.4 Comparison to Other Methods

In the last experiment, we compared *ndtrack* with NCP, *addrwatch*, and simple OpenFlow monitoring (all described in Sect. 5) in the network with topology depicted in Fig. 8. Testing was performed with MLD snooping both enabled and disabled. Note, that NCP and Simple OpenFlow yields the same result with or without MLD snooping and consequently, both gathered the same information in each run.

Each run of the experiment followed this scenario:

1. Two Linux hosts were connected to the network.
2. Host A opened a connection to host B and the hosts transferred a file in this connection.
3. Host A initiated an one-way UDP connection outside the network.
4. Host A opened a TCP session to the remote host.
5. Hosts A and B were disconnected.

During the experiment, we monitored NC of the router, and the outputs of the monitoring tools. The results are summarized in Table 4.

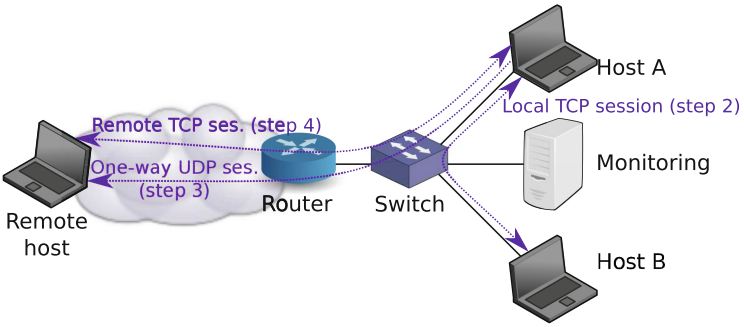

Fig. 8. Network topology used to compare our approach with older methods.

Table 4. Comparison of our approach with older methods (✓ = detected).

MLD snooping	NCP	Simple OF	addrwatch		*ndtrack*	
	Does not matter		Inactive	Active	Inactive	Active
A, B connected	-	-	✓	-	✓	✓
Local TCP	-	✓	✓	-	✓	✓
One-way UDP	-	✓	✓	-	✓	✓
Remote TCP	✓	✓	✓	-	✓	✓
A, B disconn.	-	-	-	-	✓	✓

A record for the monitored hosts appeared in the NC of the router only after a packet destined to the host arrived from the Internet. Data transfers inside the LAN and the outgoing UDP session were undetected. In addition, the record stayed in the NC (*stale* state) after the host was disconnected. Port mirroring and analysis of the traffic traversing the router would detect the outgoing UDP stream. However, local communication would be unnoticed.

While MLD snooping was disabled, *addrwatch* detected the DAD-NSes issued by the hosts when they connected to the network. Additionally, *addrwatch* reported NS messages, issued by the hosts or the router, during the data transfers. However, active MLD snooping did not leak any ND message to the monitoring computer and consequently *addrwatch* did not report any activity in the network. Moreover, *addrwatch* did not report that the hosts disconnected from the network even without MLD snooping as no ND message was sent to the network.

Both hosts were successfully identified by *ndtrack* although the tool was behind MLD snooping. In addition, *ndtrack* was able to detect that the addresses were no longer assigned.

7 Conclusions

The continuing adoption of the IPv6 protocol exposes a need for a redesign of mechanisms for user identification in LANs. Whereas in IPv4, network administrators can extract MAC and IPv4 pairings from DHCP logs, in IPv6, the pairing of IPv6 and MAC addresses is not available on a single device in the network. We studied behaviour of implementation of ND in current OSes [11]. Based on this study, we proposed a mechanism that deals with the problem of the identification of MAC and IPv6 address pairings in networks with MLD snooping both active and inactive. The proposed approach detects all address assignments in networks without MLD snooping. When MLD snooping is active, the proposed approach deals with all addresses in OpenFlow-based networks; in other networks, all addresses are detected except static addresses in FreeBSD and all addresses in OpenBSD.

The proposed approach differs from current methods in several aspects. Firstly, it is completely passive for end devices in the network. In addition, the approach does not need any modification of software or hardware used in the network. Moreover, the proposed approach detects that a new address was generated immediately without polling of active devices in the network. Furthermore, the described approach detects that an address is no longer used. Even more, the approach works for all common methods for IPv6 address distribution, namely SLAAC, stateful DHCPv6, and static assignments.

Acknowledgments. This work is a part of the project VG20102015022 supported by Ministry of the Interior of the Czech Republic. This work was also supported by the research plan MSM0021630528 and BUT project FIT-S-11-1. We would like to thank Marcela Šimková and Jim Wampler for their help during the preparation of this paper.

References

1. Dhamdhere, A., Luckie, M., Huffaker, B., Claffy, K., Elmokashfi, A., Aben, E.: Measuring the deployment of IPv6: topology, routing and performance. In: Proceedings of IMC '12, pp. 537–550. ACM, New York (2012)
2. Hinden, R., Deering, S.: IP Version 6 Addressing Architecture. RFC 4291, February 2006
3. Thomson, S., Narten, T., Jinmei, T.: IPv6 Stateless Address Autoconfiguration. RFC 4862, September 2007
4. Groat, S., Dunlop, M., Marchany, R., Tront, J.: The privacy implications of stateless IPv6 addressing. In: Proceedings of CSIIRW '10, pp. 52:1–52:4. ACM, New York (2010)
5. Narten, T., Draves, R., Krishnan, S.: Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941, September 2007
6. Narten, T., Nordmark, E., Simpson, W., Soliman, H.: Neighbor Discovery for IP version 6 (IPv6). RFC 4861, September 2007
7. Vida, R., Costa, L.: Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810, June 2004

8. Davies, J.: The Cable Guy: IPv6 Autoconfiguration in Windows Vista. TechNet Magazine, August 2007. http://technet.microsoft.com/en-us/magazine/2007.08.cableguy.aspx
9. Dunlop, M., Groat, S., Marchany, R., Tront, J.: The good, the bad, the IPv6. In: CNSR 2011, Ottawa, Canada, May 2011, pp. 77–84 (2011)
10. Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M.: Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315, July 2003
11. Polčák, L., Holkovič, M.: Behaviour of various operating systems during SLAAC, DAD, and ND (2013). http://6lab.cz/?p=1691
12. Polčák, L., Holkovič, M., Matoušek, P.: A new approach for detection of host identity in IPv6 networks. In: Proceedings of the DCNET '13, pp. 57–63. SciTePress - Science and Technology Publications (2013)
13. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. **38**, 69–74 (2008)
14. Grégr, M., Matoušek, P., Podermański, T., Švéda, M.: Practical IPv6 monitoring - challenges and techniques. In: Proceedings of IM 2011, Dublin, Ireland, pp. 660–663. IEEE CS (2011)
15. Groat, S., Dunlop, M., Marchany, R., Tront, J.: What DHCPv6 says about you. In: WorldCIS 2011, London, UK, pp. 146–151 (2011)
16. Kriukas, J.: addrwatch: A tool similar to arpwatch for IPv4/IPv6 and ethernet address pairing monitoring (2012). https://github.com/fln/addrwatch
17. Asati, R., Wing, D.: Tracking of Static/Autoconfigured IPv6 addresses. Internet Draft, version 00 (Work in progress), December 2012
18. Holkovič, M., Polčák, L.: ndtrack (2013). http://www.fit.vutbr.cz/~ipolcak/prods.php?id=308