Securing IPv6 neighbor discovery and SLAAC in access networks through SDN

Daniel Nelle, dnelle@uni-potsdam.de
Thomas Scheffler, thomas.scheffler@htw-berlin.de

ANRW 2019, Montreal, July 22nd 2019

Outline

ICMPv6

ICMPv6 is central for the deployment and use of IPv6.
It provides a number of functions that, either did not exist in IPv4, or were provided through other protocols:

- Stateless Address Autoconfiguration and Router Discovery
- Layer 2-address resolution through the ICMPv6 Neighbour Discovery Protocol (NDP) - IPv4 uses ARP.
- Determination of reachability and parameters of the transmission path: Echo Request/Response, Path MTU Discovery.
- Management of multicast group membership through Multicast Listener Discovery and Multicast Router Discovery. IPv4 uses IGMP.

Address Autoconfiguration

- Address Autoconfiguration automatically assigns IPv6 addresses to hosts. This allows hosts in a network to communicate without explicit configuration.
- There exist two methods for address configuration. Stateless autoconfiguration is the default:
  - Stateless Autoconfig: In order to build a valid IP address, the host uses available prefix information (e.g. from the Router Discovery Process) and subsequently tests this address for uniqueness, using Duplicate Address Detection (DAD).
  - DHCPv6: The stateful mechanism provides additional capabilities. DHCPv6 is not used as widely as in IPv4, still uses link-local IPv6 addresses.

Stateless Address Autoconfiguration

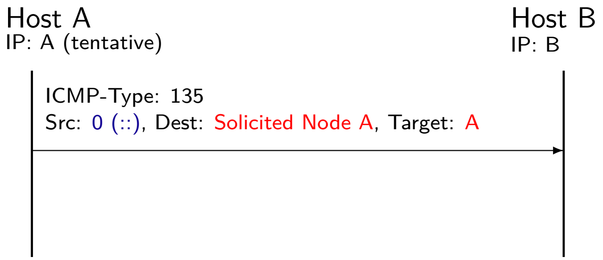
- Autoconfiguration is used only for hosts, not for routers.

- IPv6 addresses have an associated lifetime:
  - Valid addresses are preferred.
  - Addresses, that *soon* expire are in the state deprecated. Latter addresses will not be used for new connections - existing connections will continue to work.

- Duplicate Address Detection: After an address has been generated, it is in the state tentative.
- Its uniqueness on the link is verified through a Neighbour Solicitation Message. If this address already exists on the local subnet, the owner of this address sends a respective Neighbour Advertising Message and address autoconfiguration is aborted.

Stateless Address Autoconfiguration

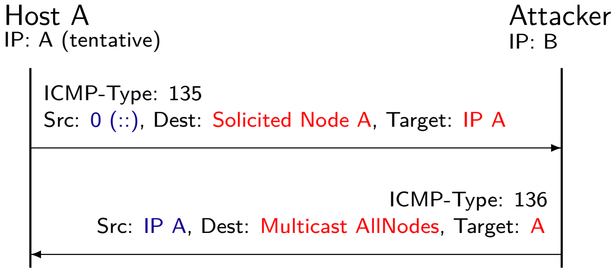
Duplicate Address Detection



- The generated IPv6 address must be unique. If is already used in the network, it must not be assigned to the interface.
- Given the large address space and the particular address generation mechanism this is a very unlikely event.

Problem statement:

- Any host can send arbitrary ICMPv6 messages!
  - Duplicate Address Detection can easily be exploited for DoS-attacks.
  - Router Advertisements and Neighbor Solicitation messages can be spoofed (think ARP-Spoofing in IPv4).


Host A
IP: A (tentative)
Attacker
IP: B
ICMP-Type: 135
Src: 0 (::), Dest: Solicited Node A, Target: IP A
ICMP-Type: 136
Src: IP A, Dest: Multicast AllNodes, Target: A

The attacker falsely and repeatedly answers the DAD-Request from Host A. Host A can not access the network.

Interworking problem between ICMPv6 and IPsec IKE

The IPv6 security architecture proposed the use of IPsec AH, for the protection of ICMPv6 traffic.

However,

there exists a Bootstrap-problem for address configuration!

- Using IKE for the establishment of a Security Association (SA) requires the hosts to have a valid IP address.
- IPsec itself is able to protect Unicast und Multicast traffic, but SAs can only be established for Unicast traffic through IKE.

  The following ICMPv6 messages would need to use manually configured SAs:

  - Router & Neighbor Solicitations
  - Router & Neighbor Advertisements

Securing Neighbor Discovery using SEND

In 2005 the SEcure Neighbor Discovery (SEND) protocol had been designed to protect against various NDP attacks (RFC 3971):

- SEND uses cryptographically generated addresses, which bind the IP address to the generating node.
- The protocol has been designed to be compatible with IPv6 address autoconfiguration.

but:

- Very limited practical experience (only marginal OS support).
- Implementations where vulnerable against resource exhaustion attacks.

How we got here...

- IPv6 NDP security issues widely discussed in 2004-2014 timeframe:
  - Active exploits: THC-IPV6-ATTACK-TOOLKIT
    `https://github.com/vanhauser-thc/thc-ipv6`
  - Router advertisement guard (RFC6105)
  - Security Implications of IPv6 Fragmentation with IPv6 ND (RFC6980)
  - ...
- Using Intrusion Detection Systems (IDS) to detect abnormal NDP traffic:
  - Implementation of IPv6 specific dissectors for Snort 2.9 (2012) & 3.0 (2019)
    `https://redmine.cs.uni-potsdam.de/projects/`

Monitoring NDP using an SDN controller

Benefits of SDN over IDS deployment:

- Works in Ethernet-switched environments.
- Potentially cheaper and easier to deploy and maintain.

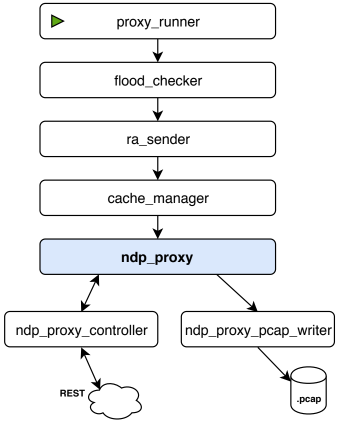
But why stop at monitoring?

- SDN controller can actively insert and filter messages.
- Steer known good traffic to fast-path, screen problematic protocols/traffic only.
- Rapid iteration and fine tuning of software-defined functions.
- Provide monitoring and alerting functionality through northbound interface.

ndp_proxy

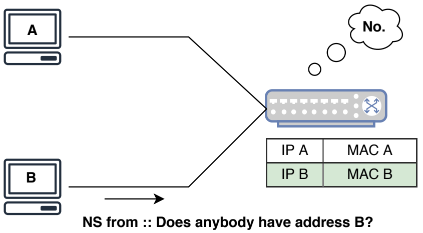
- Intelligent, selective NDP proxy using the RYU Framework
- Uses Duplicate Address Detection messages as authentication hooks
  - Creates, updates and deletes entries in a host cache
  - Interacts with NDP by dropping, forwarding or generating packets
- Acts as MAC-learning switch for IPv4
- Implements Router Advertisement Guard (RFC 6105)
- Forwards known good traffic via fast-path by inserting flow rules with different priorities:
  - *Catch all* < *Known MAC* < *Unknown IPv6* < *NDP* < *Known IPv6*
  - Manages flow rules: handles `EventOFPPacketIn` and `EventOFPFlowRemoved` events
- No changes to IPv6 protocols or host and router implementations required

Architecture

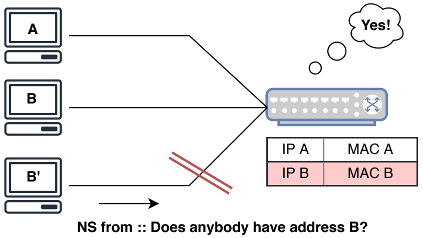


- `proxy_runner` runs app, inheritance cascade facilitates concurrent execution of modules
- `flood_checker` monitors ports by tracking requests/second
- `ra_sender` broadcasts Router Advertisements at configurable intervals
- `cache_manager` iterates over cache and triggers actions (probing, deletion, ...) if conditions are met
- `ndp_proxy` main component

Populating the Host Cache



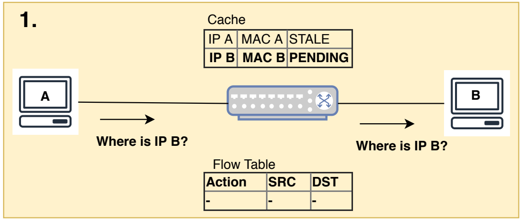
IP A	MAC A
IP B	MAC B



IP A	MAC A
IP B	MAC B

- Host B performs DUD and sends NS with source set to the unspecified address (: :)
  - If no entry for that IP exists, a cache entry is created.
  - If an entry exists, a NA is generated, causing Host B's DUD to fail. It needs to chose another address. No cache entry is created and Host B' is unable to communicate.

Handling IPv6 Neighbor Discovery

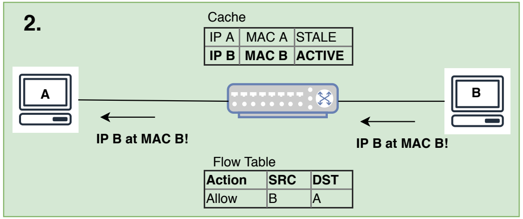


Cache

IP A	MAC A	STALE
IP B	**MAC B**	**PENDING**

Flow Table

Action	**SRC**	**DST**
-	-	-



Cache

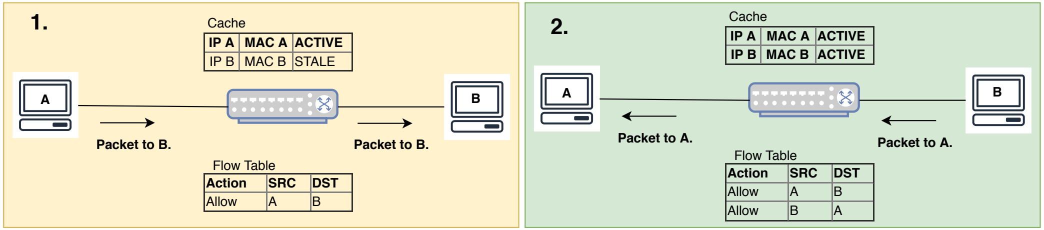
IP A	MAC A	STALE
IP B	**MAC B**	**ACTIVE**

Flow Table

Action	**SRC**	**DST**
Allow	B	A

- App checks if Host A is in cache, sets Host B's entry to pending and forwards NS.
- Host B's NA is also checked against the cache. If Host B is valid, a rule allowing communication from B to A is inserted. The cache entry is set to active and the NA is forwarded to Host A.
- The NDP process is used to keep track of activity. Entries remaining in PENDING will trigger probing before being deleted.

Regular Traffic



1. Cache

IP A	**MAC A**	**ACTIVE**
IP B	MAC B	STALE

Flow Table

Action	SRC	DST
Allow	A	B

2. Cache

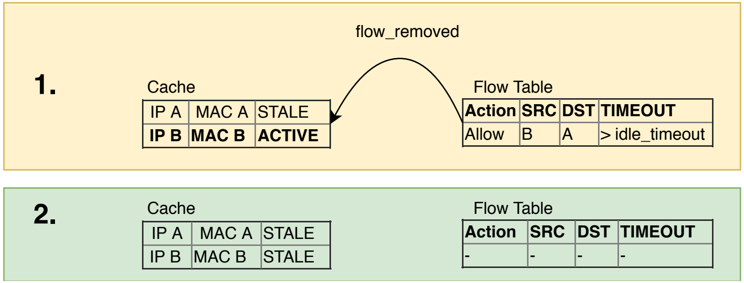
IP A	**MAC A**	**ACTIVE**
IP B	**MAC B**	**ACTIVE**

Flow Table

Action	SRC	DST
Allow	A	B
Allow	B	A

- First packet is presented to app because there is no flow rule.
- If A is in cache, packet will be forwarded, the cache entry for A set to ACTIVE and a flow rule allowing communication from A to B installed.
- Subsequent packets do not take the detour through the app.
- Same from Host B to Host A.

Flow rule timeout



Cache

IP A	MAC A	STALE
IP B	**MAC B**	**ACTIVE**

Flow Table

Action	**SRC**	**DST**	**TIMEOUT**
Allow	B	A	> idle_timeout

Cache

IP A	MAC A	STALE
IP B	MAC B	STALE

Flow Table

Action	**SRC**	**DST**	**TIMEOUT**
-	-	-	-

- Flow rules will be removed from the switch after they didn't match traffic for a configurable amount of time.
- This triggers a `EventOFPFlowRemoved`, which causes the app to set the corresponding cache entry to STALE.

Tests and Performance

Functionality tested in Mininet and using an EdgeCore AS4610-30T switch

- Onboarding IPv6 hosts works as intended
- Maintenance of valid IPv6 Host Cache is challenging

Performance:

- Controller performance tested in Mininet
- Time to process ICMPv6 message and install corresponding flow (positive tests):

Network Hosts	2	5	10	25	50	100
Latency in ms	12.39	12.98	16.32	21.63	34.62	58.21

$\rightarrow$ Host Cache data-structure is slow
Compared to ~6ms for the MAC-learning `simple_switch13` RYU implementation

Basic Assumptions and Known Limitations

Assumptions:

- One host per switch port
- Hosts are configured to perform duplicate address detection
- Controller sees and remembers all the DUD messages
- Single switch network
- Hosts are not hibernating/sleeping
- Hosts will not change their MAC addresses
  - Change to new MAC requires timeout until old entry is purged from controller host cache

Limitations:

- Reliable host state detection within Ethernets is difficult
- Hosts may be active / sleeping / crashed / disconnected
  $\rightarrow$ hosts resuming from sleep/suspend do not check for duplicate IPv6 addresses

Lessons learned and Outlook

Lessons learned:

- Ethernet SDN switches may be used for secure access network design
  - Switches are in a unique position to control network access
  - They are not (and probably never were) pure L2 devices
  - Programmable hardware provides features and speed
  - Managed WiFi already uses controller model (mostly proprietary)

Outlook:

- P4 and P4 runtime are taking over from OpenFlow?
- Reliable host state detection (active, sleeping, crashed, disconnected) $\rightarrow$ do not purge cache for active ports.
- Multiple switch network support

Contact

Email: thomas.scheffler@htw-berlin.de

WWW: https://www.htw-berlin.de

Code: https://gitup.uni-potsdam.de/dnelle/ryu_ipv6_ndp_proxy

Universität Potsdam

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences