What is wrong with the IPv6 RA protocol ?
– Some analysis and proposed solutions –

Manuel Grob, Erwin Hoffmann

FH Frankfurt am Main
April 2012

Abstract

Within IPv6 Stateless Address Autoconfiguration (**SLAAC**) the Router Advertisement Protocol plays an important rôle to provide the *Prefix* and the *Default Gateway*. We investigate the current state of the standard, the implementation on some OS and routers, investigate it's weaknesses and discuss possible solutions.

1 The Router Advertisement Protocol as part of ICMPv6

Unlike ICMP for IPv4, ICMPv6 [11] is an indispensable part of the IPv6 protocol family. ICMPv6 provides core functionalities for IPv6:

- *Stateless Address Autoconfiguration* **SLAAC** [15] by means of the mechanisms *Neighbor Discovery* and *Router Advertisement*.
- *Duplicate Address Detection* **DAD** using *Neighbor Discovery*.
- *Neighbor Unreachable Detection* **NUD**, again based on *Neighbor Discovery*.
- *Routing Support*.

These mechanisms are described within RFC 4861 [12] as part of IPv6 *Neighbor Discovery*. Here – and in the forthcoming discussion – we need to understand the dual rôle of ICMPv6 usage of *Router Solicitation/Advertisements* implementation used for *Prefix* and *Router* discovery:

1. Message container [fig. 1]
   - ICMPv6 type `133` defines the *Router Solicitation* and the
   - type `134` the *Router Advertisement* message.
2. Mechanism
   - *Router Solicitation* **RS** requested from a host for *Prefix Discovery* and answered by a (solicited) *Router Advertisement* **RA** by a router and the
   - *Unsolicited Router Advertisement* **URA** (periodically) send by router in the link segment.

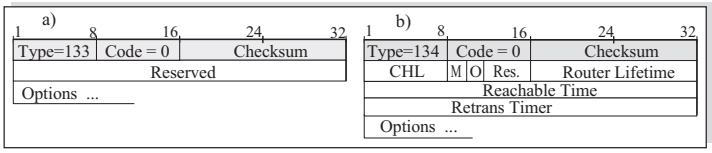

Figure 1: Structure of a) *Router Solicitation*, b) *Router Advertisement* messages; `CHL`=*Current Hop Limit*, `M`=*Managed*, `O`=*Others* flag

1.1 Options in RA messages

RA message include default information:

- The `Cur hop limit` (**CHL**), typically set to 64.
- The `Router lifetime` – in case `0` is set, the router identifies itself to be a *none-Default Gateway*.
- The `Reachable time`, and the
- `Retransmission timer`; both may be `0`.

The additional `Options` carried out in the **RA** message depend on the *mechanism*:

- `Source Link-Layer Address` (type `1`) is the most usual option and typically is set in any **RA** (and **RS**) message.
- `Prefix Discovery` employs option of type `3`. In addition to the `Prefix` and the `Prefix Length` the *lifetime* of these parameters can be provided. Furthermore, the flag `L` defines whether the prefix is (locally) available on the segment, the flag `A` tells the host to use *Stateless Address Autoconfiguration* with this prefix, whereas `M` is *managed* flag, perhaps followed by the `O` (*other*) flag.
- `Routing` (type `24`) informs the host to send any packets for this destination via this router. In order to allow load-balancing, a router may include a hint about the *priority* and route *lifetime*.
- Since IPv6 requires packet fragmentation at the sending host the maximum packet size can be announced by means of the `MTU` (type `5`) option.
- `Recursive DNS Server` (type `25`) allows the router to include the IPv6 addresses of DNS forwarders. This option is often disregarded at the host.

The options potentially provided in **RA** messages can be used to set up an IPv6 network almost entirely and relaxes the need for DHCPv6.

1.2 Prefix Discovery using Router Solicitations

In case a host is newly attached to a link segment or after the interface has been restarted, a *Router Solicitation* **RS** message is sent from that interface using the ICMPv6 message type `133` as *multicast* to the All-Router group `ff02::2` while providing it's link-local unicast (**LLU**) address `fe80::X` as source [fig. 2].

This procedure is part of the *Neighbor Discovery* **ND**. In order to support a quick *Neighbor Discovery* and to set up the *Neighbor Cache* immediately, in the *Option* field the MAC address of the sender is included in the **RS** message.

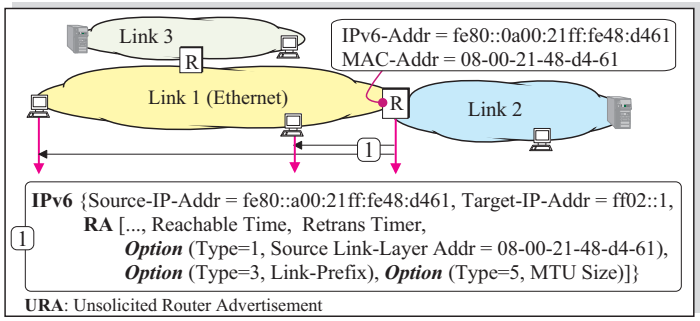

Figure 2: *Prefix Discovery* by means of *Router Solicitations*

Though RFC 4861 [12] requires a particular *validation* of the received *Router Advertisement* message (6.6.1) at the node, it is not entirely clear how the advertising router encapsulated the **RA** message in the IPv6 packet targeting the (potential) recipients. It states in section [12] 6.2.6:

> "In addition to sending periodic, unsolicited advertisements, a router sends advertisements in response to valid solicitations received on an advertising interface. A router MAY choose to unicast the response directly to the soliciting host's address

(if the solicitation's source address is not the unspecified address), but the usual case is to multicast the response to the all-nodes group."

This situation can be depict from [fig. 3] where one router employs an All-Node *multicast* in response to a *Router Solicitation* while the other has chosen to use the host's **LLU** destination address in the IPv6 packet.

1.3 Unsolicited Router Advertisements

In an IPv6 network the router attached to the local segment might not to be 'silent' but rather *multicast* their configuration to all attached nodes. Since several router might exist in a network, they may send independent *Unsolicited Router Advertisement* **URA** to the hosts.

The **RA** protocols requires to send **URA** messages not strictly periodical, but rather randomly chosen between the configured values `MinRtrAdvInterval` and `MaxRtrAdvInterval`. The advertising router needs to care about it's sending period in terms of a *state engine*.

The following situations impact sending of **URA** messages:

- The router or it's interface becomes active or a new configuration is to be processed. Now the router sends up to `MAX_INITIAL_RTR_ADVERTISEMENTS`.
- The router receives an *Router Solicitation* from an host on the segment, triggering the sending of a *Router Advertisement*.
- The router is going to be de-actived. In this case, a final *Router Advertisement* is multicasted announcing a `Router lifetime` of `0`.

2 The Operating System's view on RA

For any (*none-router*) hosts the following unfavourable situation is encountered:

- Employing **RA** messages targeting the All-Nodes `ff02::1` (upon solicitation) address is indistinguishable from a *Unsolicited Router Advertisement* since no 'address correlation' is possible at the node side.
- Even worse, according to [12] section 6.2.6, the router is required to delay the advertisement "within the range 0 through MAX_RA_DELAY_TIME". Thus, no 'time correlation' of the messages can be accomplished.
- On the other hand, a host need to be prepared to receive and process different information (i.e. *Link Prefixes*) from different routers unless the implementation of the IPv6 stack picks up the option and "MAY chose not to store of the router addresses discovered via advertisements" ([12] section 6.3.4).
- In case of contradictory information "the most recently information is considered authoritative" ([12] section 6.3.4).

In practice the IPv6 host – being targeted by (multicast) *Router Advertisements* – has little control upon the received information.

We investigate now the workaround and solutions of the major operating systems. Unlike IPv4 – being based on the initial BSD socket interface – IPv6 has three independent implementations:

1. The outcome of the WIDE/KAME [8, 18] project, which is the foundation of *BSD OS and MacOS X IPv6.
2. The Linux IPv6 implementation [13].
3. The IPv6 interface available for the Windows operating systems [9].

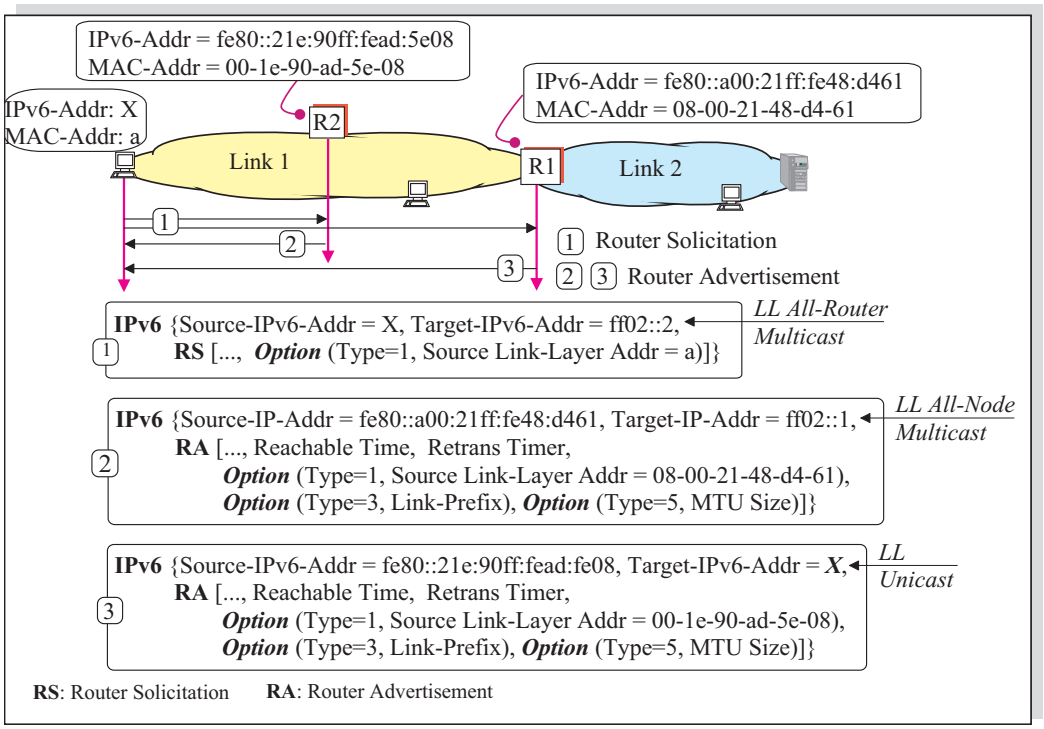

Figure 3: Multiple *Router Advertisements* on a link segment

2.1 Linux

In order to configure the IPv6 settings on a *Linux* host, one may apply the following settings:

- The standard **ifconfig** command.
- The Linux-specific **ip** command.
- The kernel configuration command **sysctl**.
- The `/proc` file system, in particular `/proc/sys/net/ipv6/`.

The Linux kernel imposes a silent concurrency limit on the received **RA** information. Only a limited number of `Prefixes` are accepted. The **RA** option `DNS Recursive Resolvers` however is ignored.

Neither the **ifconfig** nor the **ip** command can be used to configure the processing of **ND** or **RA** messages; rather this is subject of the **sysctl** kernel configurations. Linux accepts by construction **RA** messages and acts upon those accordingly.

Linux provides the following settings [lis. 1]:

```
net.ipv6.conf.all.accept_ra_defrtr=1
net.ipv6.conf.all.accept_ra_pinfo=1
net.ipv6.conf.default.accept_ra_defrtr=1
net.ipv6.conf.default.accept_ra_pinfo=1
net.ipv6.conf.lo.accept_ra_defrtr=1
net.ipv6.conf.lo.accept_ra_pinfo=1
net.ipv6.conf.eth0.accept_ra_defrtr=1
net.ipv6.conf.eth0.accept_ra_pinfo=1
net.ipv6.conf.sit0.accept_ra_defrtr=1
net.ipv6.conf.sit0.accept_ra_pinfo=1
```

Listing 1: Linux sysctl default settings

We realize that the **RA** options `Prefix` and `Routing` can be fine-tuned specifically. It is remarkable, that the 'all' and 'default' parameter settings have no influence on a specific interface. Further, it is doubtful to include

specific settings for the *loopback* interface and other *virtual* interfaces as well. Thus to disable the acceptance of any new IPv6 prefix on the Ethernet `eth0` interface, one has to set:

```
sysctl net.ipv6.conf.eth0.accept_ra_pinfo=0
```

Of course, this might be configured persistently within `/etc/sysctl.conf`.

Linux allows in addition to set IPv6 variables in the `/proc` filesystem. Since the Linux developers [4] are very creative, they have extended the understanding of the variable type BOOLEAN ...

```
/proc/sys/net/ipv6/

accept_ra - BOOLEAN
  Accept Router Advertisements;
  autoconfigure using them.

Possible values are:
  0 Do not accept Router Advertisements.
  1 Accept Router Advertisements
    if forwarding is disabled.
  2 Overrule forwarding behaviour.
    Accept Router Advertisements
    even if forwarding is enabled.

Functional default:
 enabled if local forwarding is disabled.
 disabled if local forwarding is enabled.
```

Listing 2: Linux **RA** setting according to [4]

Depending on the distribution, *prefix* and *router* might be disabled including
`IPV6_AUTOCONF=no`
into `/etc/sysconfig/network`, which will work for Red Hat and CentOS 5 Linux.

2.2 FreeBSD

IPv6 support in FreeBSD is based on the **KAME** development [8] and was available already very early – since FreeBSD 4.x. Probably, all other *BSD systems, like NetBSD, OpenBSD, and Dragonfly use the same implementation. Since FreeBSD 4.x some improvements have been included in the current FreeBSD (8.2 and 9.0).

Actually, FreeBSD offers two interfaces to tailor the IPv6 settings:

- The **ifconfig** command. Unlike it's Linux *pendant* layer 2 (**MAC**) and layer 3 (in particular **IP**) settings can be customized.
- The **sysctl** kernel configuration.

FreeBSD **sysctl** kernel interface provides a few setting only. Here we have two 'knobs' to be effective [5]:

- `sysctl net.inet6.ip6.accept_rtadv=0|1`
- `sysctl net.inet6.ip6.forwarding=0|1`

and posses the following meaning:

accept_rtadv	forwarding	role of the node
0	0	host (to be manually configured)
0	1	router
1	0	autoconfigured host (spec assumes that host has single interface only, autoconfigured host with multiple interface is out-of-scope)
1	1	invalid, or experimental (out-of-scope of spec)

Table 1: FreeBSD **sysctl** setting for **RA** messages

However, **ifconfig** allows an enhanced handling of **RA** messages per *interface* [lis. 3]:

```
accept_rtadv
    Set a flag to enable accepting
    ICMPv6 Router Advertisement messages.

-accept_rtadv
    Clear a flag accept_rtadv.
```

Listing 3: FreeBSD **RA** settings by means of **ifconfig**

Initially, FreeBSD is set up NOT to support IPv6 and need to be configured explicitly within `/etc/rc.conf`. The following scenarios are possible:

- *Global support* for IPv6 and acceptance of **RA** messages:
  `ipv6_enable="YES"`.

- *Qualified support* for IPv6 with **SLAAC** configured per *interface*:
  `ifconfig_IF_ipv6="inet6 accept_rtadv"`;
  here 'IF' denotes the name of the interface.
- *Restricted global support* for IPv6 disabling **RA** on a particular interface:
  `ipv6_enable="YES"`
  `ifconfig_IF_ipv6="inet6 -accept_rtadv"`

Thus, FreeBSD provides the same level of control for **RA** message comparable to Linux. FreeBSD restricts the amount of accepted IPv6 prefixes received by **RA** messages employing the limit `PRLSTSIZE=10`.

2.3 MacOS X

Apple's MacOS X kernel is a fork of FreeBSD 4.1x particular customized and called *Darwin* [1]. Thus, MacOS X supports IPv6 natively originating from the KAME project as well.

However, Apple did not include the enhancements of FreeBSD but rather followed their own philosophy, providing IPv6 support out-of-the-box. Customization is provided by means of the *System Preferences.app* GUI and some kernel parameters available via the **sysctl** [2] command [lis. 4]:

```
net.inet6.ip6.kame_version:
   20010528/apple-darwin
net.inet6.ip6.forwarding: 0
net.inet6.ip6.redirect: 1
net.inet6.ip6.hlim: 64
net.inet6.ip6.accept_rtadv: 0
```

Listing 4: Some MacOS X IPv6 kernel parameters

While in general autoconfiguration of the IPv6 address can be disabled by the GUI, neither the kernel, nor the **ifconfig** command allows a specific setting how to react on **RA** messages.

2.4 Windows

Starting with Windows XP, Microsoft offers IPv6 support [9] – and with Windows Vista this is the preferred protocol enabled by default – using **SLAAC** and IPv6 *privacy address extension* as well. Unlike IPv4 – which uses to a large extend the BSD socket interface – Microsoft developed their own IPv6 implementation close to the emerged IPv6 standards.

While for Linux the command **ip** exists, Windows makes the powerful **netsh** [10] tool [lis. 5] available; but lacks support for filtering and/or disabling **RA** messages in the WinXP versions.

```
netsh interface ipv6>show prefixpolicy
Active status will be queried ...

Predecessor  Label  Prefix
-----------------------------------------
          5      5  2001::/32
         10      4  ::ffff:0:0/96
         20      3  ::/96
         30      2  2002::/16
         40      1  ::/0
         50      0  ::1/128

netsh interface ipv6>show siteprefixes

Prefix                Valid until Interface
-----------------------------------------
2001:4dd0:ff00::/48   23h59m57s LAN connection
```

Listing 5: IPv6 prefix information derived from the **netsh** command

Starting with Windows Server 2008 and Windows vista, **RA** can be disabled my means of the **netsh** per *interface* [16]:

`netsh interface ipv6 set interface IFIDX routerdiscovery=disabled`

Here, `IFIDX` is the *interface index*.

Together with Microsoft's choice to accept any number of *prefixes* announced via **RA** messages until the entire memory is consumed, makes Windows vulnerable for any kind of ICMPv6 *DoS* attacks.

3 Routing Advertisement Daemons

Using a Unix system as a **RA** Router, today's choice is the *Router Advertisement Daemon* **RADVD**, which is the follower of the **RTADV** daemon (developed within the WIDE [18] project), to be found rarely only. In the next section, we discuss the merits of **RADVD** version 1.8.5 [14].

3.1 The RADVD

Upon activation, the **RADVD** tailors the *nix OS such the required IPv6 kernel settings are activated.

RADVD uses a *per-interface* configuration. Thus, for every physical and *virtual* interface attached to a link segment an unique configuration can be defined, as can be seen in the following generic sample [lis. 6]:

```
interface eth0 {
  AdvSendAdvert on;
  MinRtrAdvInterval 3;
  MaxRtrAdvInterval 10;

  prefix 2001:db8:1:0::/64 { };

  route 2001::/64 {
    AdvRoutePreference high;
    AdvRouteLifetime 3600;
  };

  RDNSS 2001:db8::1 2001:db8::2 { };

  DNSSL branch.example.com example.com { };

  clients fe80::a:b:c:d fe80::1:2:3 { };
}
```

Listing 6: Sample of the file `radvd.conf`

`radvd.conf` provides three levels of control [tab. 2]:

1. The *interfaces* the **RADVD** services.
2. **RA** *settings* to be configured for this interface.
3. Which additional *options* are advertised for a particular setting.

RADVD *Settings* *Options*	**Meaning**
AdvSendAdvert	enables sending Router Advertisements via the given interface
MinRtrAdvInterval **MaxRtrAdvInterval**	lower and upper limit for the interval an **URA** is sent
AdvDefaultLifetime	provide the number of seconds a router should be kept in a host's *default router list*, if not set to 0
AdvManagedFlag	tells the recipients hosts to use *stateful* address configuration in addition to SLAAC
UnicastOnly	hosts with IPv6 specifically included in the client option receive **RA** messages as *unicast*
clients	list of IPv6 addresses for hosts to be services with **RA** unicasts
prefix	defines the prefix provided to the clients for generating (usually) a global scoped IPv6 address
AdvOnLink	tells the host this prefix can be used for on-link determination
AdvAutonomous	permits the hosts to use **SLAAC** for IPv6 address set up
route	route information for which the router is able to route IPv6 traffic to.
AdvRouteLifetime	denotes the default life- time a route is valid
AdvRoutePreference	defines the preference for the denoted route. Thus, multiple router instances within a link segment can be set up to announce the same route with different preferences
RDNSS	defines a list of *DNS Forwarders* used for name resolutions
DNSSL	informs the nodes about the *DNS Search List*

Table 2: Some settings and options for *Neighbor Discovery* available within `radvd.conf`

It should be mentioned, that the current **RADVD** implementation supports RFC 5006 [6] and RFC 6106 [7] deploying the *DNS search suffix*, though the man page does not reflect this.

3.2 Sending Unicast RA messages

Though the current implementation of the **RADVD** supports in principal – according to RFC 4861 section 6.2.6 – sending of *unicast* **RA** messages, this mechanism is almost useless, since

- it requires to *register* the respective clients per IPv6 address,
- and therefore does by construction not support IPv6 *privacy extension*,
- does not allow a mixture of *multicast* and *unicast* **RA** messages.

In fact, if this option is turned on – but no IPv6 clients are explicitly included – **RADVD** does not respond to any *Router Solications* nor does it offer *Unsolicited Router Advertisements*, since they depend on *multicasting*.

3.3 Abuse of Router Advertisements

The simplicity of setting up a IPv6 network with *Router Advertisement Daemons* has (of course) disadvantages.

Prefix exhaustion:
Router Advertisements cause hosts with enabled *Prefix Discovery* to generate new IPv6 addresses based on the advertised *Prefix* and/or to update their routing table. This procedure is triggered by *solicited* or *unsolicited* advertisement on the (local) link segment. Any router advertisements – as we discussed – are addressed to the All-Nodes *multicast* address `ff02::1`.

A recipient node is unable to distinguish whether the sender of **RA** message is a valid router or not. Any malicious node can send *prefix information* with (in)valid data to the All-Nodes address and thereby undermining all attached nodes. In contrast, any node on the link segment *is required* to follow a router which is *advertising new prefix* information and to process the received information accordingly.

Current Microsoft Windows systems suffer from not limiting the generated IPv6 addresses triggered by router advertisements. As a result, it is possible to freeze such systems by simply send them a huge amount of router advertisements containing prefix information. Windows will try to handle all of them until resource exhaustion. To gain access to the machine again a cold reboot is necessary.

Route obfuscation:
By the virtue of the **RA** protocol routers are able to define a *route preference*. This information is used to enable the possibility setting up multiple routers within a link segment and denoting their order to the nodes.

As with the prefix information, a node is not able to determine whether the announced routing information is valid or not.

According to that, an attacker can set up a fake router trying to announce itself as *default gateway* with the highest preference. As a result, the attacker is able to capture the entire IPv6 traffic routed outside the link segment.

The 'Hackers Choice' [1] released a set of tools [3] to investigate weaknesses in the current implementations of the IPv6 protocol. A more detailed presentation about the tools can be found at 27C3's website [17].

Two of them cover the attacks listed above:

- `flood_router6` is a tool to flood the link segment with router advertisements containing prefix information. When started, the resulting attack causes a *DoS* on affected systems. All current Windows systems are impacted.

[1] http://thc.org/

- `fake_router6` sets up a router instance trying to announce itself as the default router within a link segment.

Since *Router Advertisements* are sent to the All-Node *multicast* address, all IPv6 enabled systems within a link segment are potentially vulnerable for DoS or spoofing attacks of malicious systems.

Especially router advertisements containing prefix information are able to cause a denial of service on systems not limiting the amount of IPv6 addresses for the interface.

4 Towards a common understanding and safe implementation of the RA protocol

Router Advertisement messages target two different nodes:

1. Router them-self: The standard (RFC 4861) provides an outline what any *other* router should do, receiving **RA** messages.
2. A standard host: *Re-acting* upon the reception of **RA** messages, once **SLAAC** is active.

The ICMPv6 RFCs provide some means to shelter the **RADV** router from *Router Solicitations*, but reversely usual hosts on the link segment left unprotected against *Router Advertisements*.

Unlike DHCP, where the client *requests* information, now with ICMPv6 the **RADV** router *commands* the hosts on the network and their IPv6 settings. We now propose some minor changes to the **RA** protocol to balance this situation.

4.1 Proposed corrections to RFC 4861

RFC 4681 allows **RA** messages to be sent either as (*Link Local*)

- **LL** *multicast* – to the All-Node address `ff02::1` – or
- **LL** *unicast* ('MAY') to individual hosts.

We propose the following adjustments in the way a **RADV** router acts on *Router Solicitations*:

1. *Router Advertisements* sent upon a received *Router Solicitation* SHOULD be send to the soliciting host's **LLU** address.
2. *Router Advertisements* following *Router Solicitations* SHOULD be send prompt – without delay – to the soliciting host.
3. *Router Advertisements* sent upon *Router Solicitations* don't impact the timing of periodically send unsolicited advertisements.

4.2 RA message handling at the interface

Section 2 of your proposal discussed some means to drop *Router Advertisements* messages typically at the host's interface. In fact, this disables to some extend **SLAAC** which is an undesired side effect.

However, following our proposed solution this situation can be relaxed in the following way assuming a flag `accept_rtadv`:

- `accept_rtadv = 0`: No processing of any *Router Advertisements*, thus **SLAAC** is (partially) disabled (discarding any ICMPv6 `type 134` messages).
- `accept_rtadv = 1`: Only *Unicast Router Advertisements* following a *Router Solicitation* are accepted (discarding **RA** messages with IPv6 target address `ff02::1` and including a type `3` option).

- `accept_rtadv = 2`: Any *Router Advertisements* are processed.

Due to our proposed changes, these functionalities can be easily realized by simple IPv6 address filters. In addition, a qualified handling of **RA** messages including a type `3` option should include the following mechanisms:

1. Upon sending a *Router Solicitation* for prefix discovery a state flag is set which is initialised to some value, typically four times the value of the (initial) *Round Trip Timer* **RTT**. Only *Router Advertisements* received within that period are accept; others – even targeting the node's **LLU** – are discarded.
2. Per interface a upper limit for accepted *link prefixes* shall be imposed and perhaps may be configurable by a particular variable (`accept_maxprefix = 12`).

These changes to the current implementations would greatly reduce the node's risk to be subject of obfuscating *Router Advertisements*. In fact, one might argue, that including a particular *token* (*challenge*) in the *Router Solicitation* message could even improve this situation because this token must be present in the received *Router Advertisements*. However, since the *Router Solicitation* is multicasted to the All-Router address, any potential attacker is able to pick up and to process it accordingly.

4.3 Modifications to the RADVD

In spite of the distinguished usage of *unicast* and *multicast* address for *RA* messages, we suggest the following changes to the **RADVD**:

- The *default* behaviour answering *Router Solicitations* should be changed, thus *unicast* address are used instead.
- In order to achieve backward compatibility a new *option* `MulticastRS` shall be introduced.
- It should be investigated, whether the `client` option really makes sense. If is option is valid only for *virtual* interfaces (as can be picked up from the docs) more user-friendly settings should be considered.
- However, considering the default of *unicasts* in case of *Router Solicitations* it is questionable, whether this mechanism is required at all.

4.4 Recommendations for Switch vendors and Multicast filtering

The proposed changes to the implementation of the ICMPv6 protocol will not be realised over night. However, there is the urgent demand to protect the (current existing) nodes against forged *Router Advertisements*.

The vendors of intelligent switches could greatly improve this situation applying the following logic:

- One (or perhaps several) switch port(s) are dedicated to the **RADVD** router.
- Rule 1: On those ports, ICMPv6 packets of `type 134` (*Router Advertisements*) from the attached nodes targeting the All-Node LL addresses with the corresponding MAC destination address `33-33-00-00-00-01` are accepted and forwarded.
- Rule 2: For all other ports, drop any ICMPv6 packets of `type 134` with All-Node IPv6 destination address.

Of course, this requires that the switch is able to filter Ethernet frames not only based on MAC addresses but rather requires investigating the IPv6 packet and even further the ICMPv6 messages, which might be tricky in the presence of IPv6 header extensions.

5 Summary

We investigated the current state of the ICMPv6 implementation available on contemporary Operating Systems and the *Router Advertising Daemon* **RADVD**.

We proposed a modification of RFC 4861 to balance the requirements of hosts and routers regarding *Router Solicitation*/*Router Advertisements*. These modifications can be relatively easily incorporated into the existing IPv6 stack providing backward compatibility.

Further, the vendors of network components, in particular *intelligent switches* may enhance their products to support the confinement of *Router Advertisements* traffic with respect to some dedicated ports.

References

[1] Apple. inet6(4) Mac OS X Manual Page. https://developer.apple.com/library/mac/#documentation/Darwin/Reference/Manpages/man4/inet6.4.html#//apple_ref/doc/man/4/inet6, 1999.

[2] Apple. Disabling kernel IPv6 support? http://lists.apple.com/archives/macos-x-server/2004/Jul/msg00013.html, 2004.

[3] The Hacker's Choice. CCC Camp release v1.8. http://thc.org/thc-ipv6/, 2012.

[4] Cybercity. /proc/sys/net/ipv4/* Variables:. http://www.cyberciti.biz/files/linux-kernel/Documentation/networking/ip-sysctl.txt.

[5] FreeBSD. Developer's Handbook. http://www.freebsd.org/doc/en/books/developers-handbook/ipv6.html.

[6] J. Jeong, S. Park, L. Beloeil, and S. Madanpalli. IPv6 Router Advertisement Option for DNS Configuration. http://tools.ietf.org/html/rfc5006, 2007.

[7] J. Jeong, S. Park, L. Beloeil, and S. Madanpalli. IPv6 Router Advertisement Options for DNS Configuration. http://tools.ietf.org/html/rfc6106, 2010.

[8] KAME. The KAME project. http://www.kame.net/, 1998.

[9] Microsoft. IPv6. http://technet.microsoft.com/en-us/network/bb530961, 2012.

[10] Microsoft. Using Netsh. http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/netsh.mspx?mfr=true, 2012.

[11] T. Narten, E. Normark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). http://www.ietf.org/rfc/rfc2461.txt, 1998.

[12] T. Narten, E. Normark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). http://www.ietf.org/rfc/rfc4861.txt, 2007.

[13] USAGI Project. Linux IPv6 Development Project. http://www.linux-ipv6.org/, 2006.

[14] Pekka Savola. Linux IPv6 Router Advertisement Daemon (radvd). http://www.litech.org/radvd/, 2011.

[15] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. http://www.ietf.org/rfc/rfc2462.txt, 1998.

[16] Usecurity.org. IPv6 Unsecurity – Router Advertisement is evil. http://it-unsecurity.org/2011/04/15/ipv6-unsecurity-router-advertisment-is-evil/, 2011.

[17] van Hauser. Recent advances in IPv6 insecurities. http://events.ccc.de/congress/2010/Fahrplan/events/3957.en.html, 2010.

[18] WIDE. WIDE v6 working group. http://www.v6.wide.ad.jp/, 2003.