# OPENBSD VS. IPV6

EuroBSDCon 2024
Florian Obser - florian @ openbsd.org

Wedgemount Lake, BC

## WHO AM I

- OpenBSD developer for 10+ years
  - `many files changed, 460k insertions(+), 535k deletions (-)`
  - August 2024: 0x7ff commits
- previous presentations:
  - BSDCan 2018: slaacd(8)
  - BSDCan 2019: unwind(8)
  - AsiaBSDCon 2023: Dynamic Host Configuration, please

## ONCE GREAT PEOPLE LIVED HERE... GIANTS... GODS... ONCE, BUT LONG AGO.

```
revision 1.1
date: 1999/12/08 06:50:20;  author: itojun;  state: Exp;
bring in KAME IPv6 code, dated 19991208.
replaces NRL IPv6 layer.  reuses NRL pcb layer.  no IPsec-on-v6 support.
see sys/netinet6/{TODO,IMPLEMENTATION} for more details.
```

## TIMELINE (WHICH WE ARE NOT USING)

- 2014-04-26: merge traceroute(8) and traceroute6(8)
- 2016-09-17: merge ping(8) and ping6(8)
- 2017-03-18: slaacd(8)
- 2018-07-10: rad(8)
- 2020-07-28: RFC 6724 IPv6 source address selection
- 2024-04-21: RFC 6724 Rule 5.5
- 2024-06-02: dhcp6leased

# IPV6 AT THE EDGE - THE CORE IS BORING

```
$ ifconfig iwm0 | fgrep inet6
inet6 fe80::f85c:5fff:fea7:df40%iwm0 prefixlen 64 scopeid 0x2
inet6 2001:1c00:270c:8e5:d89b:324e:5b04:4dc prefixlen 64 autoconf \
    pltime 2503 vltime 5203
inet6 2001:1c00:270c:8e5:7bb1:9841:7811:9527 prefixlen 64 deprecated autoconf tempora
    pltime 0 vltime 5203
inet6 2001:1c00:270c:8e5:a608:acdb:97cb:4f1b prefixlen 64 autoconf temporary \
    pltime 2503 vltime 5203
```

## RFC 6724

- "Default Address Selection for Internet Protocol Version 6 (IPv6)"
- Section 5: Source Address Selection
- obsoletes RFC 3484
  - do we need to do something?
- 8 straight forward rules

# RFC 6724 SOURCE ADDRESS SELECTION RULES

1. Prefer same address.
2. Prefer appropriate scope.
3. Avoid deprecated addresses.
4. ~~Prefer home addresses.~~
5. Prefer outgoing interface.
6. ~~Prefer matching label.~~
7. Prefer temporary addresses.
8. Use longest matching prefix.

## OUR RFC 3484 IMPLEMENTATION

```
/* RFC 3484 5. Rule 5: Prefer outgoing interface */
if (ia6_best->ia_ifp == oifp && ifp != oifp)
        continue;
if (ia6_best->ia_ifp != oifp && ifp == oifp)
        goto replace;

/*
 * At this point, we have two cases:
 * 1. we are looking at a non-deprecated address,
 *    and ia6_best is also non-deprecated.
 * 2. we are looking at a deprecated address,
 *    and ia6_best is also deprecated.
 * Also, we do not have to consider a case where
 * the scope of if_best is larger(smaller) than dst and
 * the scope of the current address is smaller(larger)
 * than dst. Such a case has already been covered.
 * Tiebreaking is done according to the following
 * items:
```

# OUR RFC 3484 IMPLEMENTATION

```
* - the scope comparison between the address and
*   dst (dscopecmp)
* - the scope comparison between the address and
*   ia6_best (bscopecmp)
* - if the address match dst longer than ia6_best
*   (matchcmp)
* - if the address is on the outgoing I/F (outI/F)
*
* Roughly speaking, the selection policy is
* - the most important item is scope. The same scope
*   is best. Then search for a larger scope.
*   Smaller scopes are the last resort.
* - A deprecated address is chosen only when we have
*   no address that has an enough scope, but is
*   prefered to any addresses of smaller scopes.
* - Longest address match against dst is considered
*   only for addresses that has the same scope of dst.
* - If there is no other reasons to choose one,
*   addresses on the outgoing I/F are preferred.
```

# OUR RFC 3484 IMPLEMENTATION

```
* The precise decision table is as follows:
* dscopecmp bscopecmp matchcmp outI/F | replace?
*    !equal     equal      N/A    Yes |      Yes (1)
*    !equal     equal      N/A     No |       No (2)
*    larger    larger      N/A    N/A |       No (3)
*    larger   smaller      N/A    N/A |      Yes (4)
*   smaller    larger      N/A    N/A |      Yes (5)
*   smaller   smaller      N/A    N/A |       No (6)
*     equal   smaller      N/A    N/A |      Yes (7)
*     equal    larger       (already done)
*     equal     equal   larger    N/A |      Yes (8)
*     equal     equal  smaller    N/A |       No (9)
*     equal     equal    equal    Yes |      Yes (a)
*     equal     equal    equal     No |       No (b)
*/
```

# RIP IT ALL OUT

- Brooding over this for days
- Could not find a case where this did anything
- Theory: Code and RFC drafts evolved in parallel leading to bit-rot.

```
revision 1.240
date: 2020/07/28 17:54:15;  author: florian;  state: Exp;  lines: +52 -153
Rewrite IPv6 source address selection in terms of the 8 rules given in
RFC 6724 section 5.
This simplifies the code considerably while extensive testing shows no
change in behaviour. It is time to volunteer some more testers.
OK denis@ some time ago.
```

# FINAL TIE-BREAKER

- 8 rules produce a candidate set, not a single address
- Old implementation: pick newest configured address
- New implementation: pick oldest configured address
- Has consequences for flash-renumbering events, as found by naddy

## FINAL TIE-BREAKER

- Both old and new behaviour are implementation details (TAILQ)
- When someone changes the data structure, behaviour changes
- Use highest pltime / vltime

## RULE 5.5

- There are 9 rules!
- Rule 5.5: Prefer addresses in a prefix advertised by the next-hop.
    - "Rule 5.5 is only applicable to implementations that track this information."
- Important for multi-homing small office / home office.
- Not applicable for ISPs with their own address space.
- Works in legacy-IP because of NAT

## RULE 5.5 IMPLEMENTATION

- We do a route lookup before source address selection
  - (most of the time)
- Use the p2p gateway field of `struct in6_ifaddr` to store next-hop.
- (Need to pass information from userland)

# RULE 5.5 IMPLEMENTATION

```
-in6_ifawithscope(struct ifnet *oifp, struct in6_addr *dst, u_int rdomain)
+in6_ifawithscope(struct ifnet *oifp, struct in6_addr *dst, u_int rdomain,
+    struct rtentry *rt)
[...]
+       struct in6_addr *gw6 = NULL;
+
+       if (rt) {
+               if (rt->rt_gateway != NULL &&
+                   rt->rt_gateway->sa_family == AF_INET6)
+                       gw6 = &(satosin6(rt->rt_gateway)->sin6_addr);
+       }
[...]
                         * Rule 5.5: Prefer addresses in a prefix advertised
                         * by the next-hop.
-                        * We do not track this information.
                         */
+                       if (gw6) {
+                               struct in6_addr *in6_bestgw, *in6_newgw;
+
+                               in6_bestgw = &ia6_best->ia_gwaddr.sin6_addr;
+                               in6_newgw = &ifatoia6(ifa)->ia_gwaddr.sin6_addr;
+                               if (!IN6_ARE_ADDR_EQUAL(in6_bestgw, gw6) &&
+                                   IN6_ARE_ADDR_EQUAL(in6_newgw, gw6))
+                                       goto replace;
+                       }
```

## STATELESS ADDRESS AUTO CONFIGURATION (SLAAC)

- Router sends multicast icmp6 "Router Advertisements":
    - "I'm a default router!"
    - Use this /64 to form IP addresses (yolo!)
    - (I also know about name servers)
- Host waits for Router Advertisements (or solicits them)
    - uses prefix information to form stable and temporary addresses
    - configures default route
    - (uses name server information)

## SLAAC - HOST (OLD)

- Split between kernel & userland
- Kernel listens for router advertisements, configures addresses
- Userland: rtsol(8) / rtsold(8)
    - sends router solicitations

## SLAAC - HOST (OLD)

- Kernel has to parse complicated packets
    - security issue
- Kernel did not go through ioctl(2) path
    - Awkward for kernel unlocking work
- rtsol(8) pre-dates WiFi and suspend / resume
    - Runs in one-shot mode

## SLAAC - HOST (NEW)

- Rip it all out!
- Replace it with slaacd(8)
  - Priv'seped & pledged
  - Always-on
  - Configuration: `ifconfig iwm0 inet6 autoconf`
  - Handles WiFi roaming, suspend / resume, DNS, multiple interfaces…

## SLAAC - ROUTER (OLD)

- rtadvd(8)
  - "[rtadvd.conf(5)] obeys the famous termcap(5) file format."

    ```
    default:\
        :chlim#64:raflags#0:rltime#1800:rtime#0:retrans#0:\
        :pinfoflags="la":vltime#2592000:pltime#604800:mtu#0:
    ef0:\
        :addr="2001:db8:ffff:1000::":prefixlen#64:tc=default:
    ```

  - Too old, too beige, plain needed killing.

## SLAAC - ROUTER (NEW)

- rad(8)
  - Priv'seped & pledged
  - `parse.y` based config file, de-facto standard to configure things in OpenBSD

```
dns {
    nameserver 2620:fe::fe:9
    nameserver 2620:fe::9
}
interface vlan42 {
    auto prefix # this is the default
}
interface vlan64 {
    nat64 prefix 64:ff9b::/96
}
```

## IPV6 PREFIX(ES) FOR RAD(8) - DHCPV6-PD

- Request Prefix Delegation(s) from ISP router (CPE)
- (Split up delegated prefix)
- Configure IPv6 addresses on downstream interfaces
    - rad(8) picks these up
- dhcpcd(8) from ports can do this

## DHCPV6-PD - IN BASE - PROBLEMS

- Someone needs to
  - be sufficiently bored
  - have a need
  - be able to do something about it

## DHCPV6-PD - IN BASE - STARS ALIGN

- My ISP rolls out DHCPv6
- CPE is just too crapy
  - request just the right thing
  - answers only exactly once
  - otherwise factory reset
- `want.html` to the rescue
  - Mischa & Ibsen take care of it and send me a FritzBox

## DHCPV6-PD

- dhcp6leased(8) (transmogrified dhcpleased(8))
    - Priv'seped & pledged
    - `parse.y` based config file

```
request prefix delegation on em0 for {
      vlan42
      vlan64
}
```

## DHCPV6-PD

- multiple prefixes

```
request prefix delegation on em0 for {
      vlan42
}
request prefix delegation on em0 for {
      vlan64
}
```

## DHCPV6-PD

- Config Debugging
- Split vs. multiple prefixes / forcing a prefix length

```
$ dhcp6leased -nvvf ./dhcp6leased.conf
request prefix delegation on em0 for {  # prefix length = 56
        vlan42/64       # 2001:db8::/64
        vlan64/64       # 2001:db8:0:1::/64
        reserve/57      # 2001:db8:0:80::/57
}

request prefix delegation on em0 for {  # prefix length = 64
        vlan23/64       # 2001:db8::/64
}
```

## V6-MOSTLY NETWORKS

- IPv6 misconfiguration is found and not hidden by happy eyeballs
- Those who can, do
  - Client opt-in via DHCPv4 option
  - PREF64 option in Router Advertisements
- Those who can't… still get IPv4

# 464XLAT

- NAT64 - Provider-side translator (PLAT)

```
pass in log on vlan64 inet6 from any to 64:ff9b::/96 af-to inet \
    from 192.168.178.3
```

- NAT46 - Client-side translator (CLAT)

```
pass in log quick on pair2 inet af-to inet6 \
    from 2001:db8::da68:f613:4573:4ed0 to 64:ff9b::/96 \
    rtable 0
```

## GELATOD (CLAT)

- dynamically sets up the translation rule

```
pass in log quick on pair2 inet af-to inet6 \
    from 2001:db8::da68:f613:4573:4ed0 to 64:ff9b::/96 \
    rtable 0
```

- in ports (because of complicated configuration)

```
ifconfig pair1 inet 192.0.0.4/29
ifconfig pair2 rdomain 1
ifconfig pair2 inet 192.0.0.1/29
ifconfig pair1 patch pair2
route add -host -inet default 192.0.0.1 -priority 48
```

## 464XLAT FUTURE WORK

- remove `pass in` limitation from `af-to`?
- go the macOS way?

```
inet 192.0.0.2 netmask 0xffffffff broadcast 192.0.0.2
inet6 2001:67c:370:1998:14d1:485:d69a:8641 prefixlen 64 autoconf secured
inet6 2001:67c:370:1998:c97d:f537:8e4c:bd22 prefixlen 64 autoconf temporary
inet6 2001:67c:370:1998:72:a0dc:2780:ea8f prefixlen 64 clat46
nat64 prefix 64:ff9b:: prefixlen 96n
```

# QUESTIONS?

florian @ openbsd.org