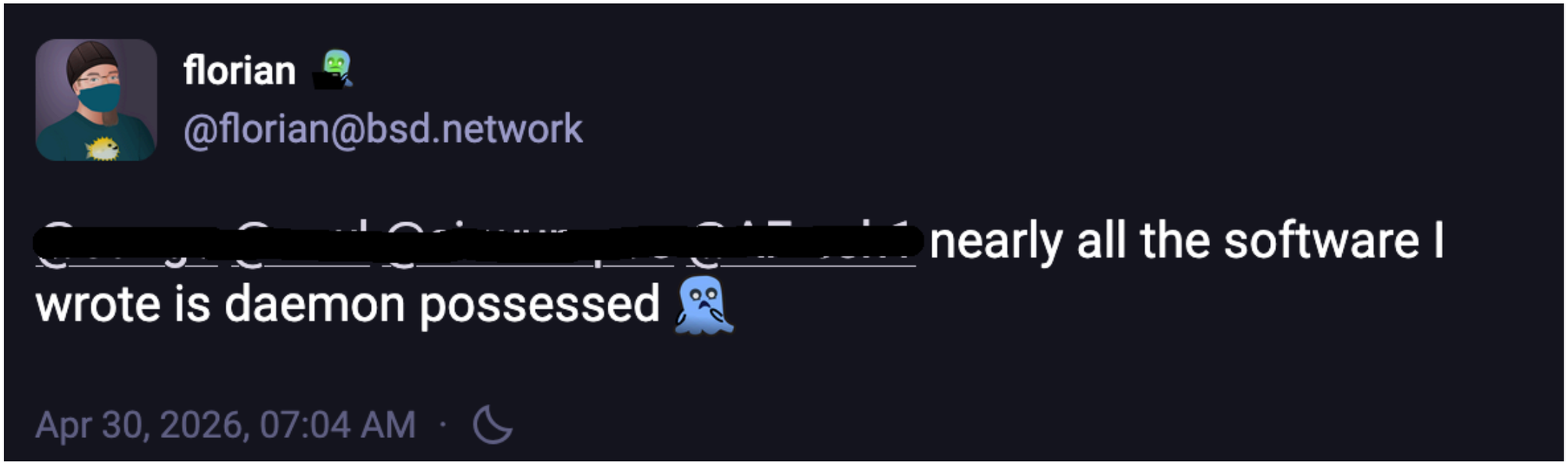
Let's find out how to get predictable IPv6 addresses assigned to OpenBSD VMs

Or: Mischa is looking for a solution...

BSD-NL Early 2026 - Florian Obser - florian @ openbsd.org

Who Am I

- OpenBSD developer for 13+ years



- to wit:
  - `dhcpleased` / `dhcp6leased` / `rad` / `slaacd` / `unwind`
- deleted 300k LoC from `usr.sbin/bind`
  - current contributions are net-negative; will retire at zero

The Problem

- How to get predictable IPv6 addresses on OpenBSD
- Smashing return in the installer:

```
$ cat /etc/hostname.vio0
inet autoconf
inet6 autoconf
```

- `ifconfig`

```
lladdr be:b4:95:d3:08:81
inet6 2001:db8::41b0:9c1d:1894:11b0 prefixlen 64 autoconf pltime 3485 vltime 7085
inet6 2001:db8::d02e:db57:6ac3:ede6 prefixlen 64 autoconf temporary pltime 3485 v
```

- autoconf: Semantically Opaque Interface Identifier (soii)
  - stable between reboots, but per prefix random IPv6 address
- temporary: completely random, doesn't make sense for servers.

Solving Problems

1. Ask an LLM (eew)

Solving Problems

1. Ask an LLM (eew)
2. Do it yourself (and maybe ask others for help)

Solving Problems

1. Ask an LLM (eew)
2. Do it yourself (and maybe ask others for help)
3. Organize a conference and nerd-snipe someone to give a presentation

autoinstall(8)

- Net boot
- Answer file is fetched from `http://$next_server/MAC_address-install.conf`
- Get example answer file by installing once by hand and check root's mailbox.

```
System hostname = openbsd-install
Network interface to configure = vio0
IPv4 address for vio0 = autoconf
IPv6 address for vio0 = 2001:db8::42
IPv6 prefix length for vio0 = 64
Network interface to configure = done
IPv6 default router = fe80::1%vio0
[...]
```

- see install.site(5) for even more control
  - per-host tgz & `/install.site` script

config management system du jour

- ansible, ~~puppet~~ OpenVox, salt…
- manage `/etc/hostname.vio0`
- or, collect IP addresses and populate DNS

disable temporary & soii for IPv6

```
cat <<END > /etc/hostname.vio0
inet autoconf
inet6 autoconf -temporary -soii
up
END
```

- EUI64 address, formed from mac address
  - `lladdr be:b4:95:d3:08:81` → `inet6 2001:db8::bcb4:95ff:fed3:881`

/128 per vlan

- `rad.conf`

```
interface vlan42 {
	no auto prefix
	prefix 2001:db8::42/128
}
```

- yes, this works with `slaacd(8)`
  - … but not much else

DHCPv6

- `dhcp6leased(8)`
- only implements DHCPv6 prefix delegation
- `dhcp6leased.conf`

```
request prefix delegation on vio0 for {
    vio1/64
}
```

`dhcp6leased(8)` Overview

1. main process (dhcp6leased)
    - parses & reloads config
    - reads, writes & parses lease file
    - spins up other processes
    - configures network
    - open UDP sockets
2. frontend
    - talks to the network & control socket
3. engine
    - tells the frontend to send a packet
    - parses answer packet (received from frontend)
    - tells main processes to configure network

A few more hints

- DHCPv6 is specified in RFC 8415.
- DHCPv6 state-machine is the same for all address options
  - Identity Association for Prefix Delegation (IA_PD)
  - Identity Association for Non-temporary Addresses (IA_NA)
  - [Identity Association for Temporary Addresses (IA_TA)]
- DHCPv6 options on-wire formats are described in chapter 21.

Your mission, should you choose to accept it…

- … transmogrify IA_PD into IA_NA.
- Remember the configuration: `request prefix delegation on vio0 for {`

```
$ git grep -i 'request prefix delegation'
dhcp6leased.conf.5:A list of interfaces on which to request prefix delegation:
dhcp6leased.conf.5:.It Ic request prefix delegation on Ar name Ic for [...]
parse.y:ia_pd           : REQUEST PREFIX DELEGATION ON STRING FOR {
printconf.c:                    printf("request prefix delegation on %s for {"
printconf.c:                    printf("request prefix delegation on %s for {\n",
```

- ~~Documentation~~
- Lexer / Parser
- ~~Print configuration for `dhcp6leased -nv`~~

parse.y

- contains lexer + yacc(1) based parser
- add rule for `"request address for INTERFACE"`

parse.y

```
ia_pd           : REQUEST PREFIX DELEGATION ON STRING FOR {
                        iface_conf = conf_get_iface($5);
                        iface_ia_conf = calloc(1, sizeof(*iface_ia_conf));
                        if (iface_ia_conf == NULL)
                                err(1, "%s: calloc", __func__);
                        iface_ia_conf->id = iface_conf->ia_count++;
                        if (iface_conf->ia_count > MAX_IA) {
                                yyerror("Too many prefix delegation requests");
                                YYERROR;
                        }
                        SIMPLEQ_INIT(&iface_ia_conf->iface_pd_list);
                        SIMPLEQ_INSERT_TAIL(&iface_conf->iface_ia_list,
                            iface_ia_conf, entry);
                } iface_block {
                        iface_conf = NULL;
                        iface_ia_conf = NULL;
                }
                ;
```

parse.y

```
ia_na           : REQUEST ADDRESS FOR STRING {
                        iface_conf = conf_get_iface($4);
                        iface_conf->ia_na = 1;
                        iface_conf = NULL;
                }
                ;
```

parse.y

- hook it up to the grammar

```
grammar         : /* empty */
                | grammar '\n'
                | grammar varset '\n'
                | grammar conf_main '\n'
                | grammar ia_pd '\n'
                | grammar error '\n'            { file->errors++; }
                ;
```

- extend lexer

```
lookup(char *s)
{
        /* This has to be sorted always. */
        static const struct keywords keywords[] = {
                {"commit",      COMMIT},
                {"delegation",  DELEGATION},
                {"for",         FOR},
                {"on",          ON},
                {"prefix",      PREFIX},
                {"rapid",       RAPID},
                {"request",     REQUEST},
        };
[...]
```

printconf.c

- Left as an exercise.
- Fill in missing bits in `printconf.c`
- Create `dhcp6leased.conf` with `request address for vio0`
- run `make obj; make; obj/dhcp6leased -nvf ./dhcp6leased.conf`

Requesting a lease

- DHCPv6 is a request - response protocol
- Statemachine for IA_NA is the same as for IA_PD
- We need to send a solicit packet → frontend.

frontend.c

```
__dead void      frontend_shutdown(void);
void             frontend_sig_handler(int, short, void *);
void             frontend_startup(void);
void             update_iface(uint32_t);
void             route_receive(int, short, void *);
void             handle_route_message(struct rt_msghdr *, struct sockaddr **);
void             get_rtaddrs(int, struct sockaddr *, struct sockaddr **);
void             udp_receive(int, short, void *);
int              get_flags(char *);
struct iface    *get_iface_by_id(uint32_t);
struct iface    *get_iface_by_name(const char *);
void             remove_iface(uint32_t);
void             set_udpsock(int, uint32_t);
void             iface_data_from_imsg(struct iface*, struct imsg_req_dhcp *);
ssize_t          build_packet(uint8_t, struct iface *, char *);
void             send_packet(uint8_t, struct iface *);
int              iface_conf_cmp(struct iface_conf *, struct iface_conf *);
```

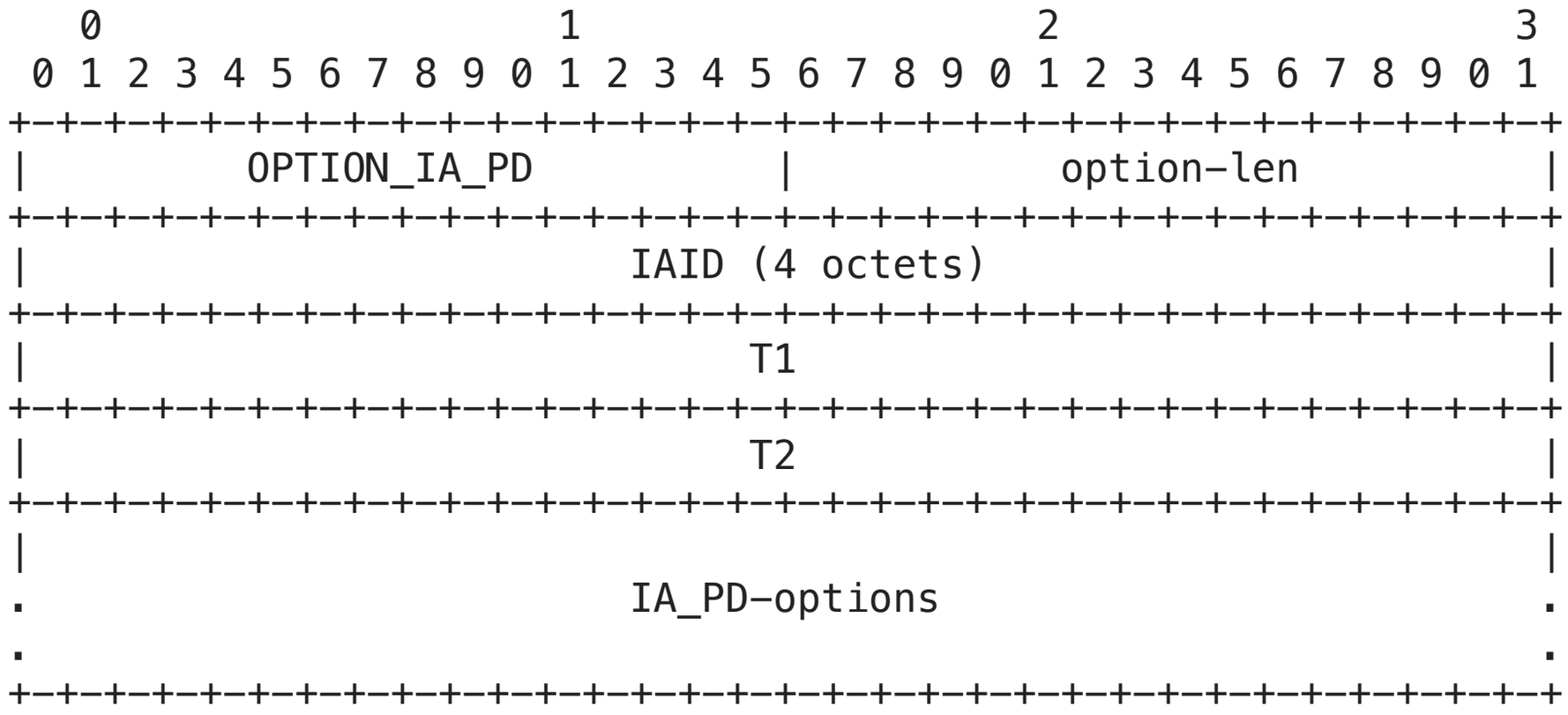
`frontend.c – build_packet`

- fills in global `dhcp_packet`
  - dhcp header
  - required options (client ID, maybe server ID)
- loops over IA_PD in `struct iface_conf`
  - adds option header
  - adds IA_PD option

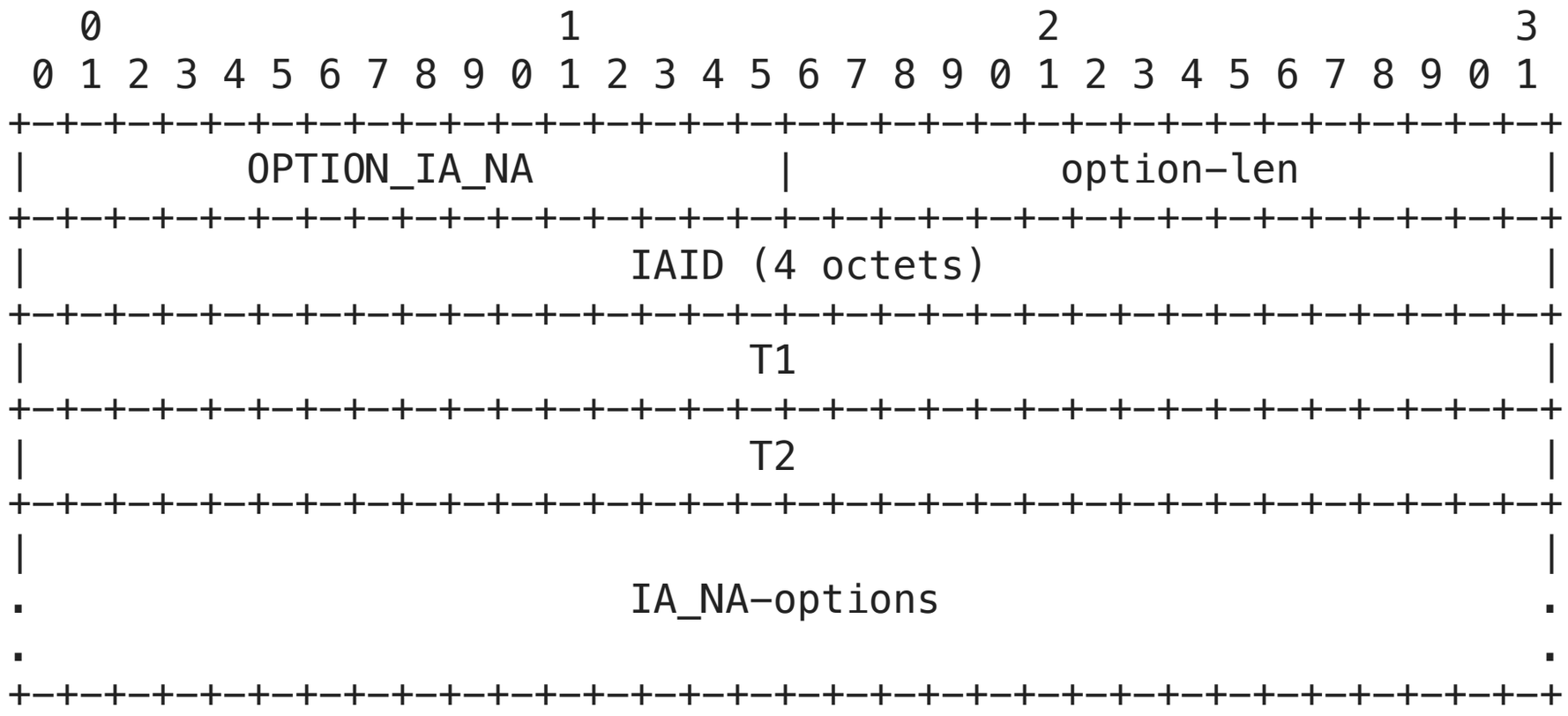
```
SIMPLEQ_FOREACH(ia_conf, &iface_conf->iface_ia_list, entry) {
        struct prefix *pd;

        opt_hdr.code = htons(DHO_IA_PD);
[...]
}
```

RFC 8415 - 21.21. Identity Association for Prefix Delegation Option



RFC 8415 - 21.4. Identity Association for Non-temporary Addresses Option



frontend.c – build_packet

- add IA_NA option

```
if (iface_conf->ia_na) {
        opt_hdr.code = htons(DHO_IA_NA);
[...]
}
```

frontend.c – build_packet

- replace `struct dhcp_iapd` with `struct dhcp_iana` (even if it's the same)
- old

```
iapd.iaid = htonl(ia_conf->id);
iapd.t1 = 0;
iapd.t2 = 0;
memcpy(p, &iapd, sizeof(struct dhcp_iapd));
p += sizeof(struct dhcp_iapd);
```

- new

```
iana.iaid = htonl(1);
iana.t1 = 0;
iana.t2 = 0;
memcpy(p, &iana, sizeof(struct dhcp_iana));
p += sizeof(struct dhcp_iana);
```

- leave out `IA_NA-options` we don't have any (yet).

Test

- run `tshark -nl -i vio0 -V -Y dhcpv6 'port 546 or port 547'`
- run `dhcp6leased -dv`
- `tshark` should show a `SOLICIT` message going out and `ADVERTISE` or `REPLY` message coming in.
- `dhcp6leased` should print "unhandled option: 3"

unhandled option: 3

- 3: `OPTION_IA_NA`

```
grep -l 'unhandled option:'
engine.c
```

- `parse_dhcp()`

```
[...]
case DHO_IA_PD:
[...]
        break;
case DHO_RAPID_COMMIT:
[...]
        break;
default:
        log_debug("unhandled option: %u", opt_hdr.code);
        break;
```

- You know the drill, transmogrify `case DHO_IA_PD` into `case DHO_IA_NA`.

Draw the rest of the owl!

- define new structs & constants
- pass address hints between engine & frontend
  - goes into `IA_NA-options`
- configure IPv6 address
  - the code is there for the PD case
- write lease & parse lease
- `dhcp6leasectl`
- everything else I forgot.

Questions?

florian @ openbsd.org