NSWI184 – Řízení počítačových sítí

Cviko prvé

Kateřina Kubecová, Ondřej Zajíček

02. října 2025

Cviko prvé

Organizace předmětu

Úvod do simulace počítačových sítí

Dynamický routing

Forma, kontakty

- přednášky a cvičení (2/2)
- website: `https://nswi184.jmq.cz/`
- Ondřej Zajíček: santiago at crfreenet dot org
- Kateřina Kubecová: katerina dot kubecova at nic dot cz

Podmínky zápočtu

Pro úspěšné započtení je nutné předvést **nasimulování a routing sítě dle vylosovaného zadání.** Losování proběhne přibližně v polovině listopadu, předvedení dle domluvy. **Není nutné chodit na cvičení, ale bude se vám to velmi hodit.**

Podmínky zkoušky

Zkouška je ústní s písemnou přípravou, **open-book a open-RFC**. Cílem je přednesené látce rozumět, nikoli ji ovládat perfektně zpaměti. Na začátku zadáme otázky, vy si rozmyslíte odpovědi, případně dohledáte v literatuře, a pak si o tom popovídáme.

Očekávaný plán semestru

1. Simulace sítí, statický routing, RIP
2. RIP a OSPF ve FRR a BIRDu
3. Pokročilé OSPF
4. Babel
5. Základní nastavování a zkoumání BGP
6. Filtrování BGP, ROA, ASPA
7. *není co cvičit*
8. BGP zběsilosti
9. Zabezpečujeme síť, ladíme problémy
10. IPv4
11. Tunely
12. Pokročilé L2+ technologie: VLAN, VxLAN, MPLS
13. Sítě pro koncové uživatele

EduHawk

Instalace

- `git clone https://gitlab.nic.cz/labs/eduhawk/`
- `cd eduhawk`
- `make install`

Manuál: `eduhawk help`

Dvojice propojených počítačů

```
eduhawk create foo    # create a cluster foo
eduhawk start foo lel # create a machine lel in cluster foo
eduhawk start foo bar # create a machine bar in cluster foo
eduhawk shell foo bar # open a shell to machine bar

ip link               # show link status
ip link set lo up     # set up loopback

# Create a link
#    on lel it's called tobar
#    on bar it's called tolel
eduhawk veth foo lel tobar bar tolel
```

Dvojice propojených počítačů

```
ip link                 # show link status
ip link set tobar up    # set the link up
ip link set tolel up

ip a                    # ip address -> show address info
ping fe80::...%tobar    # ping the other side
                        # (e.g. ping fe80::58a9:99ff:fe5d:2a9f%tolel)
```

Příkaz `ip a` zobrazuje **lokální adresy**. Pokud chceme `ping` z `lel` na `bar`, musíme pingat ze stroje `lel` adresu rozhraní `tolel` na stroji `bar`.

```
tcpdump -i any -Z root -n     # dump everything on the links
```

Nefunguje TCPdump?

Uvnitř EduHawka je zakázaný syscall `setgroups`.
Starší hardened buildy ho zavolají vždy, i když chcete `-Z root`.
Problém lze vyřešit přeložením TCPdumpu bez implicitního `-Z`.

```
git clone https://github.com/the-tcpdump-group/tcpdump.git
cd tcpdump/
./autogen.sh
./configure
make
```

Třetí počítač

```
# create a machine nit in cluster foo
eduhawk start foo nit

# and connect to lel
eduhawk veth foo lel tonit nit tolel
```

Tím máme tři propojené počítače.

- Zkuste donastavit síť tak, aby `nit` pingnul `lel`.
- Co potřebujeme, aby `nit` pingnul `bar`?

Manuální routing

```
nit <-> lel <-> bar
```

bar potřebuje globální adresu

```
ip link add self type dummy          # dummy interface to bear the "myself"
ip link set self up                  # it must be up to work
ip a add 2001:db8::1/128 dev self    # assign an address
```

Manuální routing

```
nit <-> lel <-> bar
```

bar potřebuje globální adresu

```
ip link add self type dummy          # dummy interface to bear the "myself"
ip link set self up                  # it must be up to work
ip a add 2001:db8::1/128 dev self    # assign an address
```

nit potřebuje vědět, kde je bar:

```
ip route add 2001:db8::1/128 via fe80::... dev tolel
```

Manuální routing

```
nit <-> lel <-> bar
```

bar potřebuje globální adresu

```
ip link add self type dummy          # dummy interface to bear the "myself"
ip link set self up                  # it must be up to work
ip a add 2001:db8::1/128 dev self    # assign an address
```

nit potřebuje vědět, kde je bar:

```
ip route add 2001:db8::1/128 via fe80::... dev tolel
```

```
From fe80::...%tolel icmp_seq=1 Destination unreachable: No route
```

Manuální routing

`nit <-> lel <-> bar` ... `lel` taky potřebuje vědět, kde je `bar`:

```
ip route add 2001:db8::1/128 via fe80::... dev tobar
```

Manuální routing

`nit <-> lel <-> bar` ... `lel` taky potřebuje vědět, kde je `bar`:

```
ip route add 2001:db8::1/128 via fe80::... dev tobar
```

... a že má forwardovat

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Manuální routing

`nit <-> lel <-> bar` ... `lel` taky potřebuje vědět, kde je `bar`:
`ip route add 2001:db8::1/128 via fe80::... dev tobar`

... a že má forwardovat
`echo 1 > /proc/sys/net/ipv6/conf/all/forwarding`

... oops!
```
From fe80::5851:71ff:feec:96e9%tolel icmp_seq=1
Destination unreachable: Beyond scope of source address
```

Pořád neforwardujeme z `lel` dál. **Proč? Co se děje?**

Manuální routing

`nit <-> lel <-> bar` ... `lel` taky potřebuje vědět, kde je `bar`:
`ip route add 2001:db8::1/128 via fe80::... dev tobar`

... a že má forwardovat
`echo 1 > /proc/sys/net/ipv6/conf/all/forwarding`

... oops!
```
From fe80::5851:71ff:feec:96e9%tolel icmp_seq=1
Destination unreachable: Beyond scope of source address
```

Pořád neforwardujeme z `lel` dál. **Proč? Co se děje?**

Globální adresu totiž potřebuje i zdroj, tj. `nit`.

Manuální routing

`nit <-> lel <-> bar` ... co zbývá?

- přidat globální adresu zdroje
- přidat routy směrem zpátky

Už to pingá? Zkuste `socat`!

```
socat TCP6-LISTEN:6666 STDIO
socat TCP6:[2001:db8::1]:6666 STDIO
```

Manuální routing

Přidáme si čtvrtý stroj do řady!

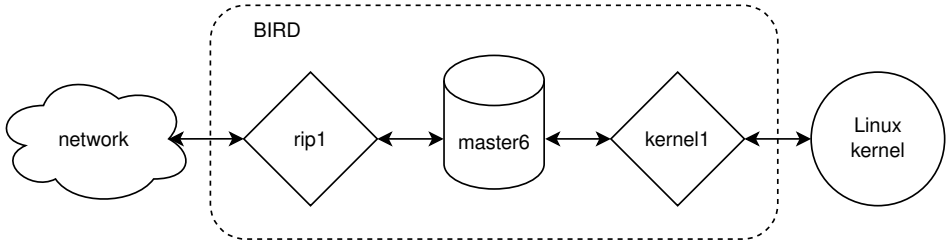
Nainstalujeme si BIRD

- `git clone https://gitlab.nic.cz/labs/bird -b stable-v3.1`
- `cd bird && autoreconf -i && ./configure && make -j?`
- `bird` → daemon, který budeme pouštět
- `birdc` → klient, kterým se k daemonu připojíme

Konfigurace BIRDa

- konfigurační soubor, obvykle `bird.conf`
  - `log "bird.log" all;` → logujeme do lokálního souboru, neobtěžujeme tím Syslog
  - `protocol device {}` → potřebujeme znát informace o síťových rozhraních
- `bird -l` → použij `./bird.conf` a vytvoř `./bird.ctl`
- `birdc -l` → připoj se na `./bird.ctl`
- zajímavé příkazy `birdc`:
  - `show status`
  - `show interfaces`
  - `show protocols`
  - `show route`
  - `disable ...`
  - `enable ...`
- Co se stane s výstupem `show interfaces`, když vypnete device protocol?

Protokol RIP v BIRDu




```
router id 42; # Na každém stroji jiné číslo!
protocol direct { ipv6; }
protocol rip ng {
  ipv6 { import all; export all; };
  interface "*" {};
}
protocol kernel {
  ipv6 { export all; };
}
```

Spuštění BIRDa v EduHawku

- Vytvoříme nový cluster (`eduhawk create fee`)
- Přidáme stroje `bar`, `lel` a `nit` (`eduhawk start fee lel ...`)
- Linky přidáme (`eduhawk veth fee lel tobar bar tolel`)
  a nahodíme (uvnitř `lel`) `ip link set tobar up`.
- Ve složce `fee` bychom měli mít podsložky `bar`, `lel` a `nit`,
  v každé z nich pak podsložku `ext`.
- Do `ext` každého stroje překopírujeme spustitelné soubory `bird` a `birdc`.
- Vytvoříme v `ext` konfigurační soubor (předchozí dva slidy)
- Otevřeme shell (`eduhawk shell fee bar`), `cd ext`,
  a spustíme BIRD (`./bird -l`).
- Zbytek vyřeší BIRD, můžeme udělat `ping`.

Protokol RIP v BIRDu

- Vyzkoušejte si `ping` mezi stroji.
- Co ukazuje `tcpdump`?
- Vypněte a zapněte RIP na jednom stroji. Jak se síť chová?
- Přidejte linku dokolečka, pozorujte `tcpdump` a `ping`; co se změní?
- Přidejte víc strojů, pozorujte `tcpdump`, vypisujte `show route`.
- Jak velkou (funkční!) síť stihnete vyrobit, než skončí cviko?