ISA - Síťové aplikace a správa sítí

Laboratorn manual

Vysoké učení technické v Brně

https://github.com/nesfit/ISA/tree/master/manual

Tento laboratorní manuál slouží jako referenční příručka pro laboratorní cvičení předmětu Síťové aplikace a správa sítí (ISA).Shrnuje potřebné prerekvizity pro práci v laboratoři a přidává doplňující informace k jednotlivým cvičením. Doporučujeme, aby se studenti s tímto materiálem seznámili v rámci přípravy na laboratorní cvičení.

Doporucene texty pro pr pravu na cvicen

- Všeobecné prerekvizity: kapitoly 1, 2, 3 a 4.
- Cvičení 1 (základy konfigurace sítě, analýza provozu): kapitoly 2, 3.
- Cvičení 2 (zabezpečený přenos dat): kapitoly 5, 6, 7.
- Cvičení 3 (DNS): kapitola 8.
- Cvičení 4 (VoIP): kapitola 12.
- Cvičení 5 (správa sítě): kapitoly 9, 10, 11.

Obsah

1 Adresovan v IP s t ch

1.1 Adresovan v s ti s protokolem IPv4

Pokud v síti provozujeme pouze komunikaci IPv4 [18], používáme IPv4 adresy pro identifikaci síťových zařízení, směrování (směrovací protokoly RIP, OSPF, EIGRP) či filtrování dat (firewally).

1.1.1 Formát IPv4 adresy

IPv4 adresa je jedinečný identifikátor síťového rozhraní, který se používá pro adresování na vrstvě IP. Délka IPv4 adresy je 32 bitů. Standardně se zapisuje v dekadickém tvaru ve formátu `X.X.X.X`, kde `X` je dekadický zápis jednotlivých bytů adresy oddělených tečkou. Příklad:

```
8.8.8.8   (dekadicky) -- 00001000 00001000 00001000 00001000 (binárně)
127.0.0.1 (dekadicky) -- 01111111 00000000 00000000 00000001 (binárně)
```

IPv4 adresa se skládá ze dvou částí:

- adresy sítě (prefix) a
- adresy uzlu.

Všechny počítače ve stejné podsíti mají stejný prefix a liší se pouze adresou uzlu. Například u sítě s prefixem 192.168.1.0/24 je možné přidělit koncovým zařízením adresy 192.168.1.1, 192.168.1.2 až 192.168.1.254. Délka prefixu 24 bitů, tj. prvních 24 bitů adres počítačů z této sítě je stejných. Protože délka prefixu může být variabilní, zadává se při konfiguraci IP adresu u počítače buď formou délky prefixu za lomítkem (např. 192.168.1.1/24) nebo formou síťové masky, kdy jedničkový bit znamená bit prefixu (např. 255.255.255.0).

Pro danou podsíť existují dvě speciální adresy, které nelze použít pro adresování koncových zařízení:

- adresa sítě - IP adresa, kde adresu uzlu tvoří samé nulové bity, např. 192.168.1.0/24
- broadcastová adresa - IP adresa, kde adresu uzlu tvoří samé jedničkové bity, např. 10.10.10.255/24

Příklad adresování s maskou /22, kde 22 bitů tvoří prefix sítě a zbývajících 10 bitů adresu uzlu:

```
                                          < ---  prefix sítě --- ><adresa uzlu>
Adresa sítě:   147.229.12.0/22     binárně: 10010011 11100101 00001100 00000000
Broadcast:     147.229.15.255/22   binárně: 10010011 11100101 00001111 11111111
Příklad:       147.229.12.101/22   binárně: 10010011 11100101 00001100 01100100
Maximální počet stanic pro adresování: 2^10 - 2 = 1022

Rozsah IPv4 adres pro adresování koncových stanic:
               147.229.12.1/22     binárně: 10010011 11100101 00001100 00000001
               147.229.12.2/22     binárně: 10010011 11100101 00001100 00000010
               ...
               147.229.12.255/22   binárně: 10010011 11100101 00001100 11111111
               147.229.13.0/22     binárně: 10010011 11100101 00001101 00000000
               147.229.13.1/22     binárně: 10010011 11100101 00001101 00000001
               ...
               147.229.14.1/22     binárně: 10010011 11100101 00001110 00000001
               ...
               147.229.15.1/22     binárně: 10010011 11100101 00001111 00000001
               ...
               147.229.15.254/22   binárně: 10010011 11100101 00001111 11111110
```

1.2 Adresovan v s ti s protokolem IPv6

Protokol IPv6 definuje pro adresování IPv6 adresy o délce 128 bitů [5]. Podobně jako u komunikace IPv4, pokud je v síti nastaven protokol IPv6, tak pro identifikaci počítačů (přesněji řečeno síťových rozhraní počítačů) se používá adresa IPv6. Tyto adresy se pak používají pro směrování (protokoly RIPng, OSPFv3, EIGRP for IPv6), filtrování dat apod.

Protože formát protokolů IPv4 a IPv6 je naprosto odlišný, není možné vytvořit spojení na IP vrstvě mezi stanicí s adresou IPv4 a stanicí s adresou IPv6. Je ale možné nakonfigurovat na počítači (i na stejném síťovém rozhraní) obě adresy, tj. jak adresu IPv4 tak i adresu IPv6 (tzv. dual stack). V takovém případě počítač komunikuje IPv4 protokolem se stanicemi, které jsou adresovány IPv4 adresou, a protokolem IPv6 se stanicemi, které mají nakonfigurovánu adresu IPv6. Protože se jedná o vrstvu IP, tak pro přenos na transportní vrstvě (protokoly TCP a UDP) či na aplikační vrstvě (např. služba HTTP) se nic nemění.

Protokoly IPv4 a IPv6 tedy komunikují v síti nezávisle na sobě, využívají odlišné směrovací protokoly i filtrovací pravidla. Zde je potřeba si uvědomit, že firewall konfigurovaný pro IPv4 se nebude aplikovat na pakety s IPv6 a naopak.

1.2.1 Formátování adres IPv6

Adresa IPv6 se na rozdíl od IPv4 adresy zapisuje v hexadecimální podobě. Jedna hexadecimální číslice reprezentuje čtyřbitové číslo, tj. k reprezentaci 128bitové adresy potřebujeme 32 hexadecimálních číslic. V adrese IPv6 se hexadecimální číslice rozdělují do skupin oddělených dvojtečkou (:). Adresu IPv6 tvoří osm čtveřic hexadecimálních číslic. Standardní formát adresy IPv6 je `X:X:X:X:X:X:X:X`, kde `X` je čtveřice hexadecimálních číslic reprezentující 16 bitů.

```
Příklady plného zápisu adresy IPv56: 2001:067c:1220:080e:00ad:6c49:759f:3267
                                     fe80:0000:0000:0000:01cb:0238:26f4:359e
                                     0000:0000:0000:0000:0000:0000:0000:0001
```

V praxi se zapisují adresy IPv6 ve zjednodušeném formátu. Standard RFC 5952 [12] definuje pro zápis adresy následující pravidla:

1. Nuly na začátku skupin se nezapisují.

   ```
   2001:067c:1220:080e:00ad:6c49:759f:3267 -> 2001:67c:1220:80e:ad:6c49:759f:3267
   fe80:0000:0000:0000:01cb:0238:26f4:359e -> fe80:0:0:0:1cb:238:26f4:359
   0000:0000:0000:0000:0000:0000:0000:0001 -> 0:0:0:0:0:0:0:1
   ```

2. Posloupnost skupin se samými nulovými číslicemi se nahrazuje znakem `::`. Protože znak `::` lze použít v adrese pouze jednou, nahrazuje se vždy ta nejdelší posloupnost nulových číslic.

   ```
   fe80:0:0:0:1cb:238:26f4:359 -> fe80::1cb:238:26f4:359  # link-local address
   2001:0:0:1:0:0:0:1          -> 2001:0:0:1::1           # global address
   0:0:0:0:0:0:0:1             -> ::1                     # loopback address
   0:0:0:0:0:0:0:0             -> ::                      # unspecified address
   ```

3. V případě více stejně dlouhých posloupností nulových číslic, nahrazuje se znakem `::` vždy ta nejlevější posloupnost.

   ```
   2001:db8:0:0:1:0:0:1 -> 2001:db8::1:0:0:1
   ```

4. Hexadecimální znaky "a"- "f"se v adrese IPv6 vždy zapisují malými písmeny.

Stejně jako u IPv4 má adresa IPv6 dvě části: *adresu sítě* (IPv6 prefix) a *adresu rozhraní* (interface ID). Délka prefixu se zapisuje jako číslo v desítkové soustavě za lomítkem za adresou IPv6 (např. `2001:67c:1220::/46`).

1.2.2 Rozdělení adres IPv6

Adresy IPv6 se rozdělují podle typu na adresy unicastové, multicastové a anycastové [10]. Dále je možné unicastové adresy rozdělit podle dosahu na lokální či globální.

- *Unicastová adresa IPv6* slouží k identifikace konkrétního síťového rozhraní. Paket s cílovou unicastovou IPv6 adresou je směrován pomocí statické směrování nebo dynamické směrování (např. protokoly RIPng, OSPFv3, EIGRP for IPv6, MP-BGP).
- *Multicastová adresa IPv6* identifikuje skupinu síťových rozhraní, které obvykle patří různým zařízením. Paket poslaný na multicastovou adresu IPv6 je doručen všem rozhraním s danou multicastovou adresou IPv6. Pro směrování multicastového paketu IPv6 se používají multicastové směrovací protokoly (např. PIM). Prefix multicastových adres je ff00::/8.

IPv6 Multicast Address Format

8 bits	4 bits	4 bits	112 bits
1111 1111	Flag	Scope	Global ID

- *Anycastová adresa IPv6* identifikuje skupinu síťových rozhraní (uzlů). Paket poslaný na anycastovou adresu je doručen na jedno z rozhraní této skupiny, které je "nejblíže"k odesilateli vzhledem k metrice směrovacího protokolu.

Formát základních typů adres IPv6 je znázorněn v tabulce 1.

Typ adresy IPv6	Prefix (bitově)	IPv6 prefix	Příklad
linková adresa (LLA)	`1111 1110 10`	`fe80::/10`	fe80::225:90ff:fec8:3f1b
Lokální adresa (ULA)	`1111 110`	`fc00::/7`	fdd3:ce44:9703:0::1/64
Globální adresa (GUA)	`001`	`2000::/3`	2001:67c:1220:8b0::93e5:b013
Multicastová adresa	`1111 1111`	`ff00::/8`	ff02::1

Tabulka 1: Základní typy adres IPv6

1.2.3 Linková adresa IPv6 (LLA, link-local address)

Tato adresa se obvykle vytváří automaticky na rozhraní stanice. Používá se pro konfiguraci stanic. Prefix adres LLA je fe80::/10, za kterým následuje 54 nulových bitů a 64bitový identifikátor rozhraní (interface ID).

Link-Local Address (LLA)

10 bits	56 bits	64 bits
1111 1110 10	0	Interface ID

1.2.4 Lokální adresa IPv6 (ULA, unique local address)

Tato adresa slouží k adresování stanic pouze v rámci podsítě (podobně jako privátní adresy IPv4). Adresy ULA nejsou globálně směrovatelné, lze je ale směrovat na lokálních směrovačích v rámci organizace. Mají definovaný prefix `fc00::/7`.

Podle standardu RFC 4193 [11] je osmý bit definován pouze pro hodnotu 1 (lokální adresa), v praxi se tedy setkáme s adresami ULA s prefixem `fd00:/8`. Dalších 40 bitů adresy ULA tvoří globálně jedinečný identifikátor (global ID) a 16bitový identifikátor podsítě (subnet ID). Zbývajících 64 bitů adresy ULA tvoří identifikátor rozhraní (interface ID).

Unique Local Address (ULA)

7 bits	1	40 bits	16 bits	64 bits
1111 110	L	Global ID	Subnet ID	Interface ID

Adresu ULA tedy tvoří unikátní prefix délky 48 bitů, 16bitový identifikátor podsítě a adresa uzlu o délce 64 bitů.

Podle standardu má být globální identifikátor celosvětově unikátní. Proto musí být generován pseudonáhodným algoritmem, který kombinuje aktuální čas (64bitovou časovou značku z protokolu NTP), 64bitový identifikátor EUI-64 odvozený z MAC adresy, hešovací algoritmus SHA-1 [11]. Příklad implementace takového generátoru je na `https://cd34.com/rfc4193/`.

1.2.5 Globální adresa IPv6 (GUA, global unicast address)

Adresy GUA jsou jsou směrovatelné v celé síti (veřejné adresy. Tvoří je n-bitový prefix pro směrování, identifikátor podsítě (subnet ID) a adresa rozhraní (interface ID).

Global Unicast Address (GUA)

3 bits	n-bits	m-bits	64 bits
001	global prefix	subnet ID	Interface ID

2 Zaklady kon gurace linuxoveho serveru

Při práci s unixovými systémy doporučujeme využívat manuálové stránky pro zjištění významu daného příkazu (aplikace) a detailního popis parametrů spuštění, viz příkaz `man <příkaz>`, např. `man ifconfig`.

2.1 Zakladn orientace v Linuxu

Většina práce v OS Linux bude probíhat v terminálovém okně. Terminál na školních PC spustíte pomocí `Alt+F2`, zde zadáte "`gnome-terminal`", případně můžete vybrat aplikaci terminálu v nabídce aplikací.

2.1.1 Unixový souborový systém

Všechny soubory v počítači jsou uloženy v souborovém systému. Ten je organizován jako invertovaný strom souborů a adresářů, kde kořenový adresář `root` je označen jako lomítko `"/"`. Kořenový adresář obvykle obsahuje standardizovaný seznam podadresářů, do kterých se ukládají systémové, aplikační i uživatelské soubory.

Některé podadresáře a jejich význam:

- `/etc` - konfigurační soubory operačního systému,
- `/bin` - systémové aplikace a příkazy
- `/sbin` - systémové aplikace a příkazy pro správce systému (uživatel `root`)
- `/usr` - uživatelské aplikace a příkazy (`/usr/bin, /usr/lib, /usr/local`)
- `/home` - domovské adresáře uživatelů
- `/var` - soubory s proměnlivým obsahem, cache paměti, logy (`/var/log`),
- `/dev` - ovladače (drivers) k periferním zařízením počítače.

2.1.2 Základní příkazy pro práci se soubory, adresáři a procesy

Některé základní příkazy pro práci v terminálu OS Linux:

- `cd` - změna adresáře,
- `ls` - zobrazení obsahu adresáře,
- `cp` - kopírování souborů,
- `mv` - přesun souborů,
- `mkdir` - vytvoření adresáře,
- `rmdir` - zrušení prázdného adresáře,
- `touch` - vytvoření prázdného souboru,
- `rm` - smazání souboru,
- `head` - zobrazení začátku souboru,
- `tail` - zobrazení konce souboru,
- `cat` - zobrazení obsahu souboru,

- `more` - výpis obsahu souboru po stránkách,
- `grep` - hledání textu v souboru,
- `chmod` - změna přístupových práv souboru,
- `ps` - výpis běžících procesů.

Mezi textové editory patří například `nano`, `vi` nebo `vim`. Tyto editory je vhodné používat k editaci konfiguračních souborů v operačním systému. Kromě řádkových textových editorů lze použít použít i grafické editory, například `gedit`.

2.1.3 Role uživatele a správce v OS

Každý uživatel operačním systému Linux má svůj jedineční identifikátor `uid` (user ID). Kromě uživatelů jsou v OS definovány i skupiny uživatelů definované identifikátor `gid` (group ID). Každý soubor je vlastněn konkrétním uživatelem patřící do určité skupiny. Přístup k souboru je omezen přístupovými právy pro vlastníka souboru, skupinu či ostatní uživatele.

Každý proces (běžící aplikace) v systému je spuštěn pod určitým uživatelem a je jednoznačně identifikován číslem procesu `pid` (process ID).

Základní role uživatelů v unixovém OS:

- `root` - správce systému: má veškerá práva a kontrolu nad systémem.
- `user` - běžný uživatel: má pouze základní správa bez možnosti zasahovat do systémového nastavení.

Při práci v operačním systému se snažíme vyhnout používání úrovně `root`. Pokud potřebujeme použít pro určité činnosti příkazy vyžadující úroveň práv `root`, můžeme využít příkaz `sudo`, který deleguje práva správce systému danému uživateli bez nutnosti přihlásit se jako uživatel `root`. Toto oprávnění je definováno v souboru `/etc/sudoers`. Příkaz s privilegovaným přístupem pak můžeme spustit příkazem `sudo command`.

V případě potřeby je možné změnit roli uživatele příkazem `su [user]`

- `su` - přepnutí na uživatele `root`.
- `su user` - přepnutí na uživatele `user`.

2.2 Sprava systemovych sluzeb

Operační systém Linux poskytuje řadu nástrojů pro správu systémových služeb. Systémové služby obvykle běží autonomně, na pozadí systému bez přímého rozhraní pro uživatele. Operační systém proto poskytuje nástroje pro jejich manipulaci. Správu operačního systému a systémových služeb v linuxových distribucích provádí aplikace `systemd` (system demon). Pro spouštění a manipulaci systémových služeb se využívá nástroj `systemctl` (system control).

Příklady použití:

- `systemctl start [služba]` – spustí službu
- `systemctl stop [služba]` – zastaví službu
- `systemctl restart [služba]` – restartuje službu
- `systemctl enable [služba]` – povolí automatické spuštění služby po startu systému
- `systemctl disable [služba]` – zakáže automatické spuštění služby po startu systému

- `systemctl status [služba]` – vypíše informace aktuálního stavu služby a několik posledních řádků systémových logů této služby. Pokud spustíme příkaz bez parametrů, vypiš informace o všech aktuálně spuštěných službách.

Jméno služby má typicky tvar `<aplikace>.service`, například `sshd.service`. Pro zobrazení logů konkrétní systémové služby lze použít příkaz `journalctl -u [služba]`. Detailní popis příkazů a jejich parametrů naleznete v manuálových stránkách.

2.3 Konfigurace síťových zařízení

V této kapitole si ukážeme několik nástrojů operačního systému Linux, které slouží pro zjišťování síťové konfigurace.

2.3.1 Zobrazení konfigurace

Základním příkazem je příkaz `ip`. Pomocí tohoto příkazu můžeme zjistit konfiguraci všech síťových rozhraní počítače (fyzických i virtuálních), obsah směrovací tabulky a jiné. Manuálové stránky pro dané možnosti zobrazíte jako `man ip <volba>`, např. `man ip link`.

- `ip address` – zobrazí adresu na všech síťových rozhraní (též příkaz `ifconfig`).
- `ip route` – zobrazí pravidla ve směrovací tabulce
- `ip link` – zobrazí konfiguraci síťových rozhraní
- `ip neighbour` – zobrazí obsah ARP záznamů (též příkaz `arp`).

Příkaz `ip` neslouží jen k výpisu konfiguraci, ale je možné jím i nastavit parametry síťového rozhraní. Další příkazy pro zobrazení síťové konfigurace:

- `netstat -rn` – zobrazí obsah směrovací tabulky
- `ss` (`sockstat`) – zobrazí čekajících spojení či navázaných síťových spojení včetně zobrazení typu transportního protokolu (parametry `-t` pro TCP, `-u` pro UDP), zdrojových a cílových adres a portů.
- `ss state ESTABLISHED -t` – zobrazí navázaná spojení nad TCP (ve stavu ESTABLISHED)

2.3.2 Testování konektivity a síťových služeb

Následující příkazy slouží k testování dostupnosti zařízení a konektivity.

- `ping <ipv4> | <hostname>` – zašle příkaz ICMP ECHO_REQUEST na dané zařízení.
- `ping6 <ipv6> | <hostname>` – zašle příkaz ICMPv6 ECHO_REQUEST na dané zařízení.
- `traceroute <ipv4> | <hostname>` – otestuje a zobrazí cestu IP datagramu k cíli.
- `traceroute6 <ipv6> | <hostname>` – otestuje a zobrazí cestu IPv6 datagramu k cíli.
- `tcptraceroute <ipv4> | <ipv6> | <hostname>` - otestuje a zobrazí cestu paketu TCP k cíli.

Pro přihlášení na vzdálený host můžeme použít dva nástroje:

- `telnet <host>` – vzdálený přístup pomocí nešifrovaného spojení
- `ssh <host>` – vzdálený přístup pomocí šifrovaného přípojení

Testování dostupných zařízení a služeb

- `nmap` - skenování aktivních stanic v síti, otevřených portů apod.
- `telnet <host> <port>` – testování dostupnosti služeb nad TCP

2.4 Konfigurační soubory

Konfigurační soubory operačního systému a služeb se většinou ukládají do adresáře `/etc`. Jednotlivé služby zde mají svůj konfigurační soubor s názvem dané služby. Konfigurace některých služeb je uložena v adresářích `"/etc/<sluzba>.d/"` (například `/etc/rsyslog.d/`).

Důležité konfigurační soubory:

- `/etc/hostname` – obsahuje doménové jméno linuxového systému.
- `/etc/hosts` – obsahuje statické mapování doménového jména na IP adresu (nevyžaduje DNS)
- `/etc/host.conf` – obsahuje konfiguraci specifickou pro DNS resolver.
- `/etc/resolv.conf` – konfigurační soubor pro DNS resolver, který obsahuje zejména odkazy na lokální DNS servery (získané z DHCP)
- `/etc/rsyslog.conf` - konfigurační soubor aplikace `rsyslog`

Podrobnosti o konfiguraci výše uvedených služeb lze získat z manuálových stránek, např. `man resolv.conf`.

2.5 Logovaní událostí

Systémové i aplikační události jsou zaznamenávány do logovacích souborů. Logovací soubory jsou obvykle uloženy v adresáři `/var/log`. Tyto soubory jsou cyklicky rotovány po určitém čase či po dosažení určité velikosti. Dobu, po kterou jsou logy udržovány, lze nastavit v konfiguračním souboru `/etc/logrotate.conf`.

Důležité logovací soubory:

- `/var/log/messages` – obsahuje globální systémové zprávy a události.
- `/var/log/auth.log` – obsahuje zprávy týkající se autentizace uživatelů.
- `./var/log/maillog` – obsahuje logovací zprávy od mailového serveru.

Tyto logovací soubory můžeme prohlížet přímo pomocí standardních zobrazovacích příkazů (`cat, more, tail -f, grep`). Zobrazení logů většinou vyžaduje přístupová práva správce systému.

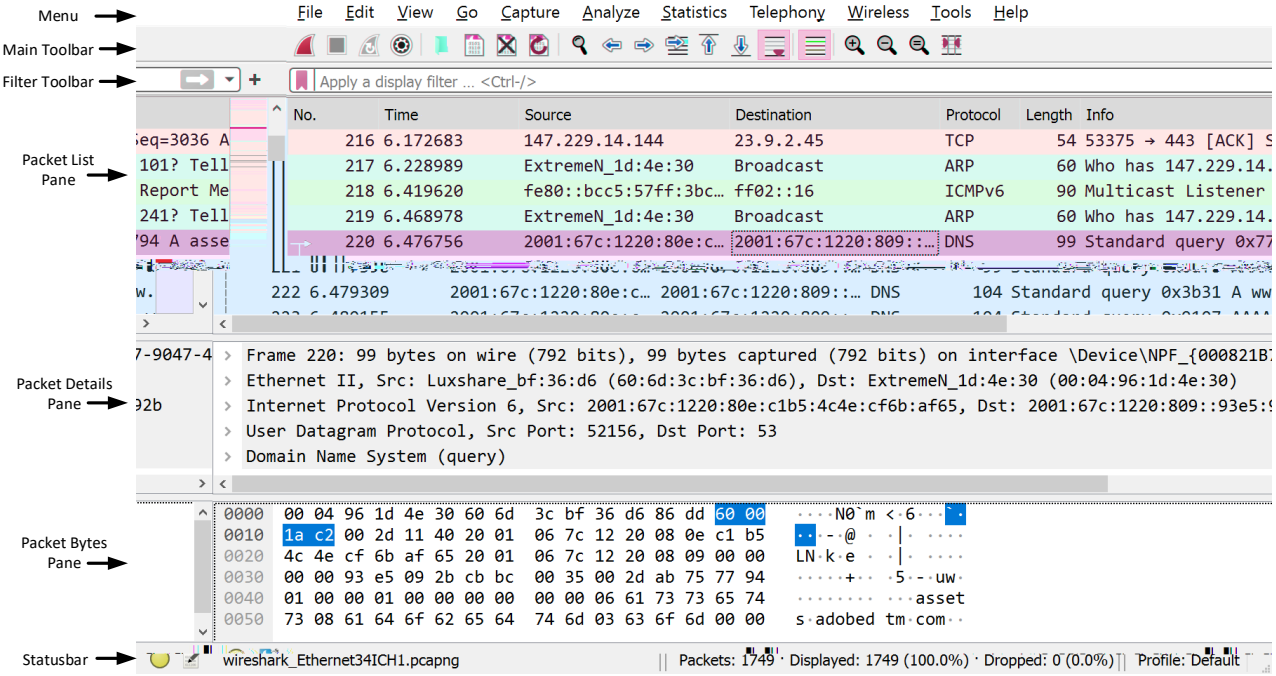
Systémové události můžeme také prohlížet pomocí nástroje `journalctl`, který umožňuje podrobněji procházet seznam zpráv, filtrovat a vyhledávat.

Příklad použití `journalctl`:

- `journalctl -n <x>` - zobrazí posledních x záznamů.
- `journalctl -p <priority>` - zobrazí pouze záznamy s danou prioritou Syslog.
- `journalctl -u <unit> | <pattern>` - zobrazí pouze záznamy s danou službou nebo obsahující zadaný řetězec
- `journalctl -t <syslog_id> | <pattern>` – vyhledává záznamy obsahující zadané ID syslogu nebo záznamy obsahující v ID syslogu zadaný řetězec.
- `journalctl -f` - průběžně tiskne na terminál poslední přidávané události
- `journalctl --since <whenstart> --until <whenend>` - zobrazí záznamy v daném časovém rozmezí.

3 Síťový analyzator Wireshark

Wireshark je grafická aplikace sloužící k zachytávání paketů a jejich analýze, viz obrázek 1. Wireshark umí zachytávat provoz na lokálních fyzických či virtuálních síťových rozhraních počítače nebo zpracovávat zachycené pakety uložené v souboru typu PCAP. V případě zpracování příliš velkých souborů (stovky MB data) je vhodnější využívat pro zpracování řádkovou verzi nástroje Wireshark, která se nazývá `tshark`[1]. Další populární nástroj pro odchytávání paketů a jejich zpracování je unixová aplikace `tcpdump`.



Obrázek 1: Wireshark

3.1 Zachytavan provozu (Packet Capturing)

Následující kroky popisují proces zachytávání paketů v programu Wireshark:

1. Zachytávat provoz můžeme z vybraného síťového rozhraní nebo z více rozhraní najednou. Rozhraní vybereme v menu `Capture -> Options`.

2. V nastavení Wiresharku je možné nastavit filtr pro výběr zachytávaných paketů (Capture Filter), pomocí kterého řekneme aplikaci, jaké pakety má zachytávat. Protože zpracování každého paketu je výpočetně náročné (dochází k parserování jednotlivých vrstev paketu a detekci protokolů), snížíme použití vstupního filtru výpočetní nároky na odchyt paketů. Políčko `Capture Filter` umožňuje vybrat již předdefinovaný filtr nebo je zadat vlastní filtr, viz `man pcap-filter`.

3. Záchyt spustíme pomocí tlačítka `Start`.

4. Odchycený provoz (tj. jednotlivé pakety spolu s časovými značkami) můžeme také uložit do souboru typu PCAP pro pozdější analýzu.

[1] Viz `https://www.wireshark.org/docs/man-pages/tshark.html`

3.2 Vyber zachyceneho provozu pomoc zobrazovac ho ltru (Display Filter)

Zachycené a zpracované pakety jsou zobrazeny v okně `Packet List Pane`. Protože nás často zajímá pouze určitý provoz, např. komunikace HTTP s konkrétní stanicí, je vhodné využít zobrazovací filtr (Display Filter), který se nachází pod nástrojovou lištou. Pomocí tohoto filtru vybereme pouze pakety s požadovanou vlastností, např. filtr `ip.addr == 147.229.8.12` vybere veškeré pakety, které mají danou zdrojovou či cílovou IP adresu.

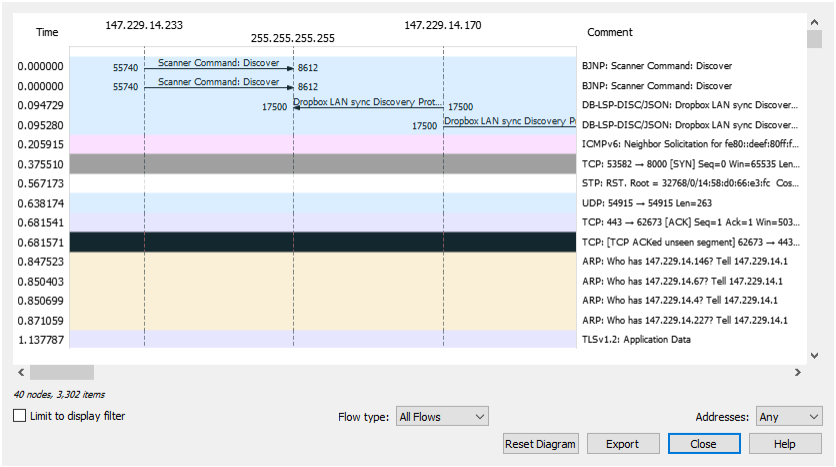
Základní operace pro vytváření zobrazovacího filtru je popsána v tabulce 2.

Porovnávání	`==, >=, <=, !=, contains`
Logické operátory	`||, or, &&, and, !, not`
Kombinace filtrů	`(ip.src==192.168.0.105 and udp.port==53)`
Filtrování na základě existence pole	`http.cookie or http.set_cookie`
Filtrování specifických bytů	`eth.src[4:2]==22:1b`
Použití regulárních výrazů	`http.host && !http.host matches ".com"`

Tabulka 2: Syntax zobrazovacího filtru (Display Filter)

3.3 Zobrazen toku (Flow Graph)

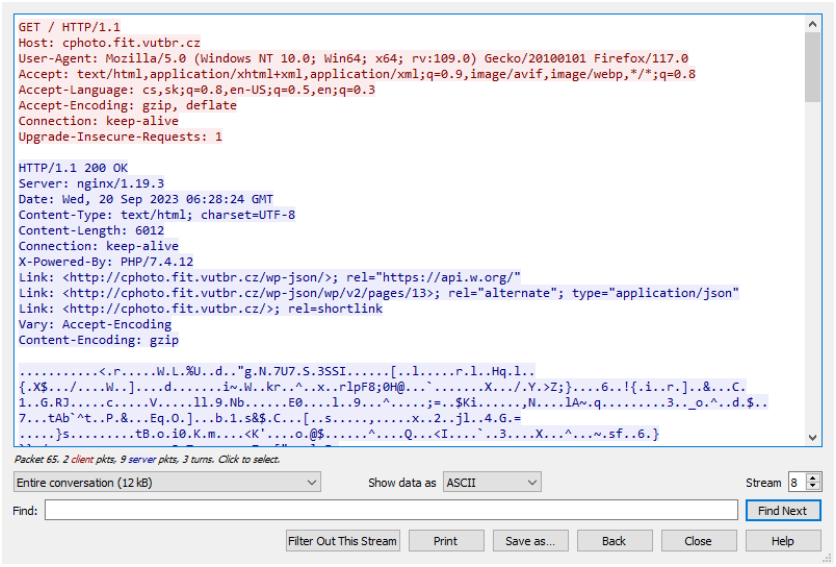
Wireshark umožňuje zobrazit časovou posloupnost komunikace pomocí tzv. grafu toků zvaného `flow graph`, viz menu `Statistics -> Flow Graph`. Graf toků zobrazuje komunikaci mezi jednotlivými zařízeními definovanými IP adresou, viz obrázek 2. Pomocí grafu toků můžeme vidět, jak často spolu stanice komunikují, jaké zprávy a v jakém pořadí si posílají a další.



Obrázek 2: Flow Graph

3.4 Zobrazen obsahu toku (Stream Analysis)

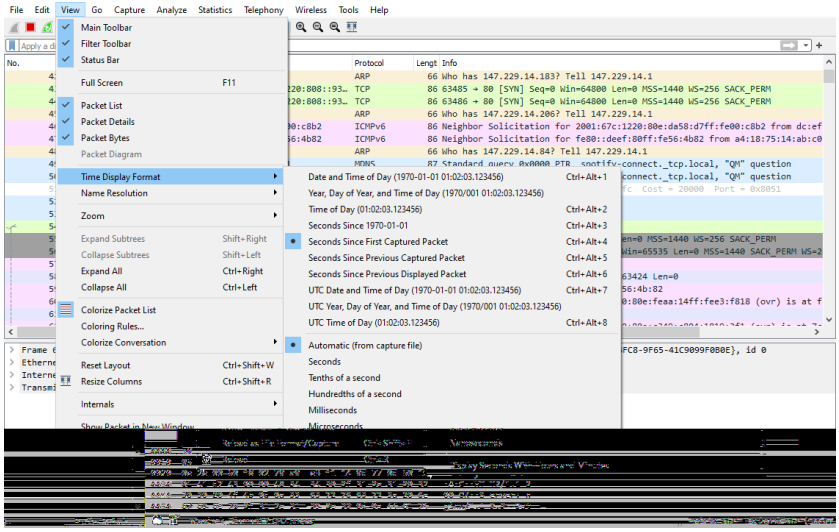
Wireshark zachytává jednotlivé pakety. Pokud chceme sledovat průběh celé komunikace včetně obsahu, můžeme zobrazit danou komunikaci (streamu) pomocí volby `Flow <protokol> Stream`, např. TCP či UDP spojení, HTTP spojení apod. Zobrazení toku provedeme tak, že vybereme jeden zachycený paket daného toku, klikneme na něj pravým tlačítkem myši a vybereme volbu `Follow <protokol> Stream`. Příklad obsahu HTTP komunikace je na obrázku 3.



Obrázek 3: Zobrazení toku HTTP

3.5 Casove znacky

Každý paket obsahuje časovou značku svého záchytu. Zobrazení časové značky závisí na nastavení Wiresharku. Wireshark umožňuje zobrazit čas jako časovou značku obsahující datum a čas záchytu, absolutní čas, relativní čas od začátku záchytu komunikace a další. Formát času můžeme nastavit v menu `View -> Time Display Format`, viz obrázek 4. Pro některé účely je vhodnější vidět pouze relativní čas od začátku záchytu paketů, pro jiné je potřeba znát absolutní čas záchytu.



Obrázek 4: Nastavení formátu časových značek

4 Konfigurace síťových koncových zařízení

4.1 Manuální konfigurace IP adres

Pro manuální konfiguraci IP adres v operačním systému Linux je možné využít příkaz `ip`:

- `ip link`, `ip addr` – zobrazí aktuální konfiguraci síťových rozhraní
- `ip link set <rozhraní> up/down` – zapne/vypne síťového rozhraní
- `ip addr add/del <IP adresa>/<prefix> dev <rozhraní>` – přidá/odstraní IP adresy
- `ip addr flush dev [rozhraní]` – odstraní všechny IP adresy z daného rozhraní
- `ip route` – zobrazí směrovací tabulku na počítači
- `ip route add default via [IP adresa]` – nastaví výchozí bránu (default gateway)

Manuální nastavení IP adres pomocí příkazu `ip` platí do nejbližšího restartu systému. Pro trvalé nastavení IP adres je potřeba použít grafické rozhraní `Network Manager`, které je dostupné přes menu *Settings/Network*.

4.2 Dynamická konfigurace IPv4 pomocí DHCP

Pro dynamickou konfiguraci IP adres je potřeba, aby byl na síti dostupný server DHCP. Pomocí protokolu DHCP provádí nejen konfiguraci IP adres, ale nabízí klientům DHCP řadu dalších parametrů, které jsou potřeba ke konfiguraci daného zařízení, například adresy serverů DNS, WINS apod.

Na cvičeních budeme používat implementace DHCP serveru a klienta zvanou ICS DHCP (Internet Systems Consortium DHCP). Pro konfiguraci DHCP serveru se používá konfigurační soubor `/etc/dhcp/dhcpd.conf` a systémová služba `isc-dhcp-server.service`.

Příklad nastavení rozsahu přidělovaných adres v konfiguračním souboru serveru DHCP:

```
Formát:
 option domain-name-servers < seznam IP adres DNS serverů>;
 subnet <IP adresa sítě> netmask <maska> {
    range <první IP adresa rozsahu> <poslední IP adresa rozsahu>;
 }
Příklad:
  option domain-name-servers ns1.isc.org, ns2.isc.org; # lokální DNS servery
  subnet 204.254.239.0 netmask 255.255.255.224 {       # podsíť
      option routers 204.254.239.1;                    # výchozí brána
      option domain-name "test.isc.org";               # implicitní doména DNS
      default-lease-time 120;                          # doba platnosti IP adresy
      range 204.254.239.10 204.254.239.30;             # rozsah přidělovaných IP adres
  }
```

Podrobnější informace o konfiguračních parametrech serveru DHCP lze najít v manuálových stránkách `man dhcpd.conf`, případně `man dhcpd`.

Jako klienta DHCP budeme používat aplikaci `dhclient`. Úkolem klienta DHCP je požádat protokolem DHCP o konfigurační parametry dané stanice (přesněji řečeno síťového rozhraní). Klient může být spuštěn pro konkrétní síťové rozhraní nebo na všech rozhraních. Přes rozhraní, na kterém je klient spuštěn, posílá dotazy serveru DHCP. Pokud nevíme, na kterém rozhraní server DHCP pracuje, je vhodné pustit klienta bez parametrů, kdy pošle dotaz s žádostí o konfiguraci na všechna aktivní síťová rozhraní.

```
dhclient -v [rozhraní]
```

Argument `-v` vypíše detailní informace o průběhu konfigurace.

4.3 Dynamicka kon gurace IPv6

Dynamická konfigurace protokolu IPv6 probíhá zcela jiným způsobem, než konfigurace DHCP u sítí s protokolem IPv4. Využívají se tři základní přístupy:

- **Autokonfigurace (SLAAC, Stateless Autoconfiguration)** [22] – autokonfigurace slouží ke konfiguraci síťových rozhraní IPv6 bez použití serveru DHCPv6. Při autokonfiguraci si připojený klient sám vytvoří adresu IPv6 z prefixu sítě (prvních 64 bitů) a identifikátoru rozhraní (spodních 64 bitů). Prefix sítě získá stanice zasláním zprávy ICMPv6 RS (Router Solicitation) na multicastovou adresu všech směrovačů v síti. Směrovač odpoví zprávou ICMPv6 RA (Router Advertisement), která obsahuje prefix sítě, L2 adresu směrovače, případně další parametry. Z adresy IPv6 odesilatele získá klientská stanice adresu výchozí brány, z odpovědi RA prefix sítě a adresu rozhraní (spodních 64 bitů adresy IPv6) si vygeneruje ze své MAC adresy mechanismem EUI-64 (rozšíření MAC adresy na 64 bitů) či si vygeneruje 64 bitů náhodně.
- **Bezstavový server DHCPv6 (Stateless DHCPv6)** [17] – v tomto případě získá klientská stanice prefix sítě a výchozí bránu ze zprávy Router Advertisement jako u autokonfigurace. Bezstavový server DHCPv6 v tomto případě neslouží k přidělení IP adresy, ale k zaslání dalších konfiguračních parametrů, např. seznamu serverů DNS, domény DNS a další. Klientská stanice tedy pošle na server DHCPv6 (s využitím multicastové adresy) dotaz typu `Information-request`, ve kterém uvede požadované parametry pro konfiguraci. Pokud je v síti nainstalován bezstavový server DHCPv6, zprávou `Reply` pošle požadované nastavení.
- **Stavový server DHCPv6 (Stateful DHCPv6)** [17] – klient posílá žádost `Solicit` pro vyhledání serveru DHCPv6. Jakýkoliv server DHCPv6 odpoví zprávou `Advertise`, ve které uvede nabízenou adresu IPv6, dobu platnosti a ostatní konfigurační parametry (servery DNS, doménu DNS). Klient odpoví žádostí `Request`, ve které uvede identifikátor serveru, se kterým komunikuje, a konfigurační parametry, které požaduje. Server potvrdí přidělení parametrů zprávou `Reply`.

Na počítačích v laboratoři lze využít službu `radvd.service`, která vysílá zprávy Router Advertisement (RA). Aplikaci je možno nakonfigurovat v souboru `/etc/radvd.conf`.

Příklad konfigurace:

```
interface <rozhraní>
{
    AdvSendAdvert on;
    MaxRtrAdvInterval <Max počet sekund mezi zprávami RA, min 4>;
    prefix <prefix>/<délka prefixu>
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };
};
```

Přehled všech možností konfigurace poskytnou manuálové stránky `man radvd` a `man radvd.conf`.

Pro zasílání zpráv RA je potřeba povolit směrování IPv6 provozu:

```
sysctl net.ipv6.conf.all.forwarding=1
```

Součásti balíčku radvd je i aplikace `radvdump`, která slouží k analýze zpráva RA: naslouchá na síťových rozhraních a zobrazuje zachycené zprávy na standardní výstup.

5 Synchronizace casu protokolem NTP (Network Time Protocol)

Protokol NTP [14] umožňuje synchronizovat čas mezi uzly v síti. Jeho úkolem je přenášet časové informace z primárních časových serverů s přesnými hodinami na ostatní servery a klienty. Protokol NTP při synchronizaci bere v úvahu proměnlivou dobou přenosu paketu, zpoždění paketu a další parametry. NTP organizuje servery hierarchicky do úrovní: primární servery NTP předávají časové informace sekundárním serverům, ty je pak předávají serverům nižší úrovně atd. Úroveň vzdálenosti od přesného zdroje času se vyjadřuje veličinou *stratum*. Nejnižší hodnota 0 označuje referenční zdroj přesného času (např. GPS). Stratum 1 označuje primární servery, které jsou synchronizovány právě s referenčním zdrojem. Stratum 2 jsou servery synchronizovány se servery stratum 1 atd. [15].

Implementaci protokolu NTP zajišťuje balík aplikací *ntp*. Tento balík se skládá z několika aplikací. V rámci laboratorního cvičení budeme používat aplikace `ntpd` či `ntpq` (viz část 5.2). Chrony[2] nabízí alternativní implementaci NTP.

5.1 Aplikace `ntpd`

Aplikace `ntpd` (NTP démon) slouží k synchronizaci času pomocí NTP. Může pracovat v režimu klienta i serveru. Aplikace běží na pozadí a neustále synchronizuje čas s nastavenými servery, případně upravuje lokální čas. Úprava lokálního času se provádí úpravou rychlosti běhu lokálního času. Pokud je lokální čas pozadu, resp. se předbíhá, tak se systémové hodiny *zrychlují*, resp. *zpomalují*. Tento způsob úpravy času znamená, že pokud je čas odchýlen o několik minut, bude nějakou dobu trvat, než dojde k synchronizaci. Na druhou stranu se tak zabrání skokové změně a navíc čas se nikdy neposune do minulosti. Pokud je lokální čas odchýlen o více než 1000 sekund (necelých 17 minut), aplikace to vyhodnotí jako chybu a skončí. Pokud se tak stane, objeví se zpráva v systémovém logu.

Zda aplikace běží, lze zjistit například příkazem `ntpq -p`. Pokud není démon `ntpd` spuštěn, skončí aplikace hláškou *ntpq: read: Connection refused*.

Službu NTP je možné spustit příkazem:

```
systemctl start ntp.service
```

Aby se předešlo problému v případě, kdy např. hardwarové hodiny jdou špatně a při vypnutí může dojít k odchýlení lokálního času o více než 17 minut, je možné vynutit okamžité nastavení času příkazem

```
ntpd -qg  # -g vypne kontrolu odchylky a nastaví čas bez ohledu na velikost odchylky
          # -q ukončí běh démona poté, co byl čas nastaven
```

5.1.1 Konfigurace

Základním konfiguračním souborem je `/etc/ntp.conf`. Tento konfigurační soubor obsahuje mnoho konfiguračních voleb. Nastavení serverů NTP zajišťuje volba `server`, za níž následuje adresa nebo jméno NTP serveru. Tato volba se může vyskytovat v konfiguračním souboru opakovaně. Pomocí této volby se aplikace `ntpd` přepne do role klienta, který může využít k synchronizaci svého času více serverů NTP. V režimu klient se lokální hodiny synchronizují podle času ze vzdáleného serveru NTP.

Kromě režimu klient může `ntpd` pracovat v režimu *peer*, která umožňuje, aby se vzdálený server synchronizoval podle lokálních hodin. To je užitečné v případě, že je v síti více serverů NTP, které mohou mít lepší zdroj času. Zároveň to řeší problém výpadků serverů.

V konfiguraci můžeme také zvolit možnosti komunikace typu *broadcast* nebo *manycastclient*, za nimiž následuje broadcastová či multicastová adresa. Na tyto adresy server NTP posílá periodické zprávy s aktuálním stavem hodin. Pro multicastové přenosy se využívají skupiny definované IANA pro protokol NTP, tj. 224.0.1.1 pro IPv4 a ff05::101 pro IPv6.

[2] Viz `https://chrony.tuxfamily.org/`.

Jinou užitečnou volbou je *restrict*, která slouží pro řízení přístupu. Aplikací `ntpd` mohou být zaslány různé požadavky přes síť (například pomocí `ntpq`). Je vhodné povolit některé dotazy jen z určitého uzlu nebo podsítě. Využití této volby může být také užitečné v případě, že se pro nastavování času využívá broadcastu nebo multicastu a není žádoucí, aby takto vyslanou informaci klienti akceptovali z libovolného zdroje. Základní tvar tohoto příkazu je:

```
restrict <adresa> [mask <maska>] [<jeden či více příznaků>]
```

Příznak definuje omezení pro danou adresu/síť:

- ignore – zahazovat všechny pakety
- nomodify – povolí pouze dotazy, požadavky měnící stav serveru jsou zahazovány
- noquery – zakáže dotazy pomocí `ntpq`, synchronizace času není ovlivněna
- notrust – zahazovat neautentizované pakety

Další příznaky a jejich popis lze nalézt v manuálových stránkách `man ntp.conf`.

5.2 Aplikace `ntpq`

Aplikace `ntpq` (NTP query) se používá pro zasílání dotazů serveru NTP `ntpd` z důvodu sledování běhu a výkonnosti, případně zjišťování aktuálního stavu. `ntpq` zasílá zprávy přes síťové rozhraní, i když běží lokálně či vzdáleně. Tato aplikace může měnit konfiguraci `ntpd` za běhu. Možnost změny parametru přes síť může být nežádoucí, proto je vhodné omezit pomocí volby *restrict* přístup pouze přes rozhraní `loopback`.

Příklad výstupu volání `ntpq -p`:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+rhino.cis.vutbr 248.205.243.78   3 u  425 1024  377   10.070   -4.280  10.575
+tik.cesnet.cz   195.113.144.238  2 u  622 1024  377    4.796   -3.464  16.528
*tak.cesnet.cz   .GPS.            1 u  364 1024  377    5.008   -3.625   6.228
```

Výpis obsahuje následující hodnoty:

remote Klient se synchronizuje vůči třem serverům: rhino.cis.vutbr.cz, tik.cesnet.cz a tak.cesnet.cz. Hvězdičkou je označený primární zdroj času, plusem sekundární zdroje času.

refid Primární zdroj času pro vzdálený server NTP.

stratum Počet skoků od vzdáleného serveru k přesnému zdroji času (1 znamená, že zdroj přesného času je přímo připojen, 16 znamená, že vzdálený server je nedosažitelný).

when Počet sekund od posledního kontaktu se serverem.

poll Počet sekund mezi jednotlivými dotazy protokolem NTP.

reachability Úspěšnost posledním 8 pokusů o kontakt vzdáleného serveru v osmičkové soustavě. (1 znamená pouze poslední pokus uspěl, 377 znamená všech 8 posledních pokusů uspělo).

delay, offset, jitter Zpoždění, posun a jitter – charakteristiky vzdáleného zdroje času, podle kterého se volí primární a sekundární zdroje času.

5.3 Aplikace pro spravu casu `chronyd` a `chronyc`

Pro sychronizaci času pomocí protokolu NTP lze využít službu `chronyd`, který běží na pozadí, provádí měření a přizpůsobení lokalních hodin síťovému času. V konfiguračním souboru `/etc/chrony.conf` lze nastavit například adresu serveru NTP, ze kterého se bude číst síťový čas. Pro práci s časem lze využít řádkového klienta `chronyc`, který zobrazuje stav programu `chronyd`, stav synchronizace času apod.

Pro práci se službou `chronyd` můžeme využít následující příkazy:

```
systemctl status chronyd.service    # zjištění stavu služby
systemctl restart chronyd.service   # restartování služby
```

Konfiguraci vzdálených serverů NTP můžeme nastavit v konfiguračním souboru `/etc/chrony.conf`, viz `man chrony.conf`. Vzdálený server NTP lze nastavit v sekci `server` nebo `pool`. Pomocí directivy `server` většinou nastavujeme synchronizaci oproti jednomu serveru NTP, zatímco v případě directivy `pool` specifikujeme množinu serverů NTP jako doménové jméno, které se resolvuje na více IP adres, které se navíc mohou v čase měnit (např. z důvodu distribuce zátěže).

Příklad konfigurace:

```
pool pool.ntp.org iburst maxsources 3
          # adresa serveru NTP je pool.ntp.org
          # parametr iburst zajistí rychlou počáteční synchronizaci
          # parametr maxsources definuje max. počet serverů,
          # které se mohou použít ze zadané množiny serverů NTP
```

Pro výpis nakonfigurovaných serverů NTP lze použít aplikaci `chronyc`, viz následující příklad:

```
210 Number of sources = 3
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#* GPS0                          0   4   377    11   -479ns[ -621ns] +/-  134ns
^? foo.example.net              2   6   377    23   -923us[ -924us] +/-   43ms
^+ bar.example.net              1   6   377    21  -2629us[-2619us] +/-   86ms
```

Tento výpis obsahuje seznam serverů NTP, se kterými služba `chronyd` komunikuje. Pro zjištění, odkud z kterého serveru se bere aktuální čas, můžeme využít příznaky `M` a `S`.

- Režim synchronizace `M` (Mode)

  Tento příznak nám říká, zda se jedná o režim `server` (`^`), režim `peer` (`=`) nebo lokální zdroj času (`#`).

- Stav zdroje času `S` (Status)

  Tento příznak definuje stav využití daného zdroje. Hodnota (`*`) označuje zdroj, podle kterého se služba `chronyd` aktuálně synchronizuje. Příznak (`+`) označuje použitelné zdroje hodin, které lze kombinovat s vybraným zdrojem. Hodnota (`-`) označuje použitelné zdroje hodin, které jsou vyloučeny ze synchronizace. Hodnota (`?`) označuje zdroje hodin, kde došlo ke ztrátě spojení. Hodnota (`x`) označuje falešné hodiny, tj. hodiny, které jsou nekonzistentní s ostatními zdroji času. Hodnota (`~`) označuje hodiny s příliš velkou proměnlivostí (nestabilní).

6 Vzdaleny pr stup pomoc SSH (Secure Shell)

Základní funkcí protokolu Secure Shell (SSH) [23] je vytvořit bezpečný přístup ke vzdálenému počítači přes nezabezpečenou síť. Díky obecnému návrhu protokolu SSH, lze pomocí něj zabezpečit i další služby, jako je např. X Window, přístup ke vzdálenému souborovému systému (SFTP, sshfs, scp), tunelování portů TCP formou tunelování aplikačních dat skrze protokol SSH . Protokol SSH zajišťuje šifrování dat, autentizaci, integritu dat a volitelně také kompresi. V předmětu ISA budeme protokol SSH využívat pouze k připojení na vzdálený terminál pomocí klíčů.

Jednou z nejčastěji používaných implementací protokolu SSH je sada programů OpenSSH[3]. OpenSSH obsahuje kromě klienta `ssh` i serverovou aplikaci `sshd`, program pro generování SSH klíčů `ssh-keygen`, agenta pro usnadnění práce s klíči `ssh-agent` a další. Podrobnosti ohledně konfigurace serveru lze najít v manuálových stránkách `sshd(8)` a `sshd_config(5)`.

6.1 Pripojen ke vzdalenemu poc taci

Pro připojení se ke vzdálenému počítači se jménem *h01* je možné použít příkaz:

```
ssh h01
```

Zde jsou nejčastěji používané parametry příkazu `ssh`, podrobnosti viz `man ssh`:

- `-l` přihlašovací jméno na vzdáleném počítači, např. `ssh -l xnovak02 merlin`.

  Varianta přihlášení přes SSH na vzdálený počítač jako uživatel `xnovak02`: `ssh xnovak02@merlin`.
- `-p` zadání nestandardního portu ssh serveru, např. `ssh -p 6222 merlin`.
- `-v` výpis podrobností ohledně přihlašování (verbose mode), např. `ssh -v merlin`.
- `-X` tunelování komunikace protokolu X11 s X serverem přes SSH, např. `ssh -X xnovak02@merlin`.
- `-L` přesměrování komunikace na daném lokálním portu na vzdálený port přes SSH, např `ssh -N -f -L 25:localhost:25 user@mailserver.example.com`, které vytvoří zabezpečené spojení z lokálního portu 25 na vzdálený port 25. Pro vytvoření tunelu se využije uživatelský účet `user` na vzdáleném mailserveru. Po dobu vytvoření tohoto spojení bude veškerý provoz z lokálního portu 25 tunelován na vzdálený port 25 přes SSH.

Po připojení ke vzdálenému serveru jsou informace o použitém spojení dostupné na serveru v proměnných prostředí typu SSH. Lze je zobrazit příkazem `env | grep SSH`, viz následující ukázka

```
local $ ssh merlin6.fit.vutbr.cz                          # připojení na server merlin
merlin $ env | grep SSH                                   # výpis proměnných
SSH_CLIENT=2001:67c:1220:80c:e138:4d11:c04c:c675 54514 22 # informace o klientovi
SSH_TTY=/dev/pts/30                                       # lokální terminál
SSH_CONNECTION=2001:67c:1220:80c:e138:4d11:c04c:c675 \    # informace o spojení
  54514 2001:67c:1220:8b0::93e5:b013 22
```

Z výpisu vidíme, že klient SSH s adresou IPv6 `2001:67c:1220:80c:e138:4d11:c04c:c675` se připojil na adresu SSH serveru s adresou IPv6 `2001:67c:1220:8b0::93e5:b013`. Klientská aplikace běžela na portu 54514 a server na standardním portu 22. Po připojení využíval vzdálený uživatel lokální terminál s identifikátorem 30.

[3] Viz `https://www.openssh.com/`.

6.2 Kop rovan souboru mezi poc taci

Pro kopírování souborů protokolem SSH lze využít příkaz `scp` s formátem

```
# kopírování lokálního souboru na vzdálený server
  scp <local-file> <username>@<ssh-server>:<path/remote-file>

# kopírování vzdáleného souboru na lokální server
  scp <username>@<ssh-server>:<path/remote-file> <local-file>
```

Následující příkaz zkopíruje lokální soubor `isa.txt` na vzdálený počítač `h01` do adresáře f̂it umístěném v domovském adresáři uživatele `student`.

```
scp isa.txt student@h01:~/fit/
```

6.3 Spusten pr kazu na vzdalenem poc taci

Protokol SSH umožňuje také spouštět příkaza na vzdáleném serveru. Je to možné zadat na straně klient SSH i serveru SSH.

- Spuštění příkazu ze strany klienta SSH

  V tomto případě zadáváme příkaz a jeho parametry jako poslední argument příkazu `ssh`. Protokol SSH zajistí přihlášení na vzdálený počítač, ale místo otevření terminálového okna pouze vykoná zadaný příkaz, zobrazí výsledek, a ukončí se. Jako příkaz lze zadat i seznam příkazů oddělených středníkem, případně skript, který se vykoná.

  ```
  Formát:
    ssh user@remote_host <command> <args>
  Příklad:                  # date -R zobrazí aktuální čas ve formátu RFC 2822
    ssh merlin.fit.vutbr.cz date -R
    Enter passphrase for key '/home/xnovak02/.ssh/id_rsa':
    Thu, 12 Oct 2023 13:26:59 +0200
  ```

- Spuštění příkazu ze strany serveru SSH (omezení přístupu, vnucený příkaz).

  Ve druhém případě se zadaný příkaz spouští na serveru místo standardního vzdáleného přihlášení. Používá se spíše k omezení přístupu pro daného uživatele, který po přihlášení pouze obdrží výsledek předem zadaného příkazu. Tento příkaz je vložen do souboru s autorizovanými klíče `authorized_keys` před vlastní klíč za klíčové slovo `command = "cmd"`.

  ```
  Soubor authorized_keys na serveru merlin:
    command="date" ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQESN
    uhsEU5GJA/+CnBaK56x0s9WSeYCqEkyfNIJibO7yZSBkW/526QzPrt16Nu
    fTJs+HtSekQJpjSoQroMtuvnG9mR4rG//0uQOU6Zdfx4ZqnEaOzRhT4XXM
    HwcU/o5UAJT9f4P/L+DItgfEoQ524HBx xnovak02@eva.fit.vutbr.cz

  Příklad použití:
    ssh merlin
    Enter passphrase for key '/home/xnovak02/.ssh/id_rsa':
    Thu Oct 12 13:40:41 CEST 2023
    Connection to merlin.fit.vutbr.cz closed.
  ```

6.4 Tunelovan provozu pres SSH

Pomocí protokolu SSH je možné tunelovat provoz z daného zařízení na vzdálený počítač. Tunelování SSH zabezpečí původně nešifrovaný provoz, který je vložen do šifrované komunikace SSH. Následující příkaz otevře spojení SSH, které pak můžeme použít jako proxy ve webovém prohlížeči.

```
ssh -D 12345 -N student@sshserver
```

Po otevření tohoto tunelu SSH stačí v prohlížeči zadat nastavení proxy serveru `localhost` s portem 12345. Provoz pak bude přesměrován na lokální počítač, poté půjde spojením SSH na `sshserver`, kde bude převeden na běžný HTTP/HTTPS provoz a poslán dále.

6.5 Generovan kl cu SSH

Pro autentizaci využívá SSH asymetrickou kryptografii, konkrétně veřejné a soukromé klíče (tzv. Public Key Infrastructure, PKI). SSH klíč se generuje příkazem `ssh-keygen`. Po spuštění bez parametrů je vytvořen pár klíčů (veřejný a soukromý klíč) algoritmem RSA o délce 2048 B. Uživatel je dotázán na umístění souborů s klíči v souborovém systému (standardně adresář `/.ssh`. Dále zadává uživatel heslo (passphrase), které chrání přístup k soukromému klíči, tzn. použití tohoto klíče. Parametr `-t` nastavuje typ šifrovacího algoritmu (implicitně RSA, může být DSA, EC DSA a další). Parametr `-b` specifikuje délku klíče (počet bitů klíče), pro RSA je implicitní hodnota nyní 3072. Parametr `-f` specifikuje jméno souboru, do kterého se klíč ukládá, parametr `-C` přidává komentář ke klíči. Podrobnější informace je možno nalézt v manuálové stránce `ssh-keygen(1)`.

Následující příklad vygeneruje klíč o délce 4096 B, který nebude chráněn žádným heslem. Klíč bude uložen v souboru `.ssh/nopass`:

```
ssh-keygen -t rsa -b 4096 -N "" -f ~/.ssh/nopass -C nopass
```

Po vytvoření klíčů vzniknou dva soubory. Klíč v souboru bez přípony je soukromý klíč, který by měl zůstat utajený a uživatel by ho neměl dále distribuovat. V souboru s příponou `.pub` je veřejný klíč, který slouží k ověřování a je určen k zveřejnění[4].

6.6 Pouzit kl cu SSH

Pro správné použití zabezpečení SSH je potřeba vygenerovaný veřejný klíč umístit na vzdálený počítač, aby tento vzdálený počítač mohl správně ověřit připojujícího se uživatel. Veřejný klíč v souboru se jménem např. `id_rsa.pub` je potřeba přenést na vzdálených počítač a uložit (přidat) do souboru `~/.ssh/authorized_keys`, kde ho bude hledat server SSH při pokusu o přihlášení.

Na vzdálený počítač můžeme veřejný klíč přenést programem `scp`. Do souboru `authorized_keys` ho můžeme přidat příkazem `cat`, viz následující příklad. Je potřeba striktně dodržovat názvy souborů – pokud dojde k překlepu ve jménu souboru (např. `authorized_key`), server SSH klíče nenajde a nebude provádět autentizaci vůči klíčům SSH.

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

Jiný způsob přenesení klíče je pomocí utility `ssh-copy-id`, viz `man ssh-copy-id`, která zkopíruje veřejné klíče z lokálního počítač na vzdálený počítač do souboru `/.ssh/authorized_keys`. Na rozdíl od předchozího řešení je možné tento přenos provést jediným příkazem, tj. není nutné přenést soubor pomocí `scp` a pak ho ručně přidat do souboru `authorized_keys`.

[4]Popis používání klíčů je například na stránce na `https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process`.

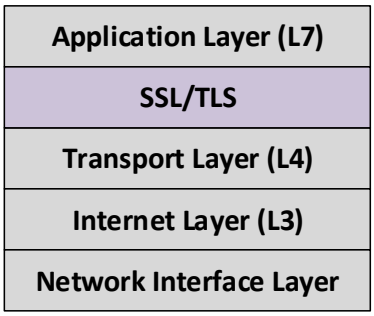
```
Formát:
   ssh-copy-id -i <key-file> [user@]hostname1] [user@hostname2] ...
Příklad:
   ssh-copy-id -i ~/.ssh/id_rsa.pub user@h23
```

Nyní se již můžeme připojit ke vzdálenému počítači pomocí vytvořených klíčů. Pokud je soukromý klíč SSH na lokálním počítači umístěn ve výchozím nastavení, použije se automaticky. Pokud jsme zvolili jiné umístění, je potřeba program `ssh` informovat o umístění soukromého klíče parametrem `-i`, nebo volbou `IdentityFile` v konfiguračním souboru. Pořadí hledání a použití klíčů u klienta SSH lze vypsat parametrem `-v`, např. `ssh -v merlin`.

7 Zabezpecen TLS (Transport Layer Security)

Původní přenosové protokoly na počátky Internetu byly nešifrované. Požadavek na zabezpečení dat se objevil později v souvislosti se zveřejněním informací o masivním odposlechu internetových linek bezpečnostními službami na Wikileaks. S tím souvisí i následná snaha o ochranu dat a soukromí u veřejných služeb jako jsou DNS, Telnet, HTTP, SMTP, FTP, VoIP a další. Přestože standardy SSL (Secure Sockets Layer) vznikaly postupně od roku 1995 a standardy TLS od roku 1999 [6], jejich rozšíření spadá do let 2014-2018, kdy například šifrování webového provozu se z cca 50% dostává na více jako 80%[5].

Protože velká část původních internetových protokolů šifrování nepodporovala (např. DNS, HTTP, SMTP, IMAP, FTP), začal se využívat koncept tunelování aplikačních protokolů přes protokoly SSL/TLS, která tak vytvářejí mezivrstvu mezi transportní a aplikační vrstvou, viz obrázek 5. Tato vrstva zajišťuje nejen šifrování, ale i autentizaci, integritu dat, bezpečnou výměnu klíčů a další funkce. Výhodou tohoto přístupu je, že přidání podpory SSL/TLS je pro existující aplikace transparentní, tzn. není nutné provádět jakoukoliv změnu stávajících protokolů.



Obrázek 5: Rozšíření modelu TCP/IP o vrstvu SSL/TLS.

Původní proprietární protokol SSL (navržený firmou Netscape Corporation) byl postupně nahrazen protokolem TLS [1]. Aktuální doporučená verze protokolu TLS je 1.3 [19][6].

TLS pracuje na principu klient – server a vytváří obousměrný zabezpečený kanál pro komunikaci na aplikační vrstvě mezi klientem a serverem. TLS zajišťuje následující typy zabezpečení:

- *Autentizace* – vždy ověřuje identitu serveru (tzv. jednostranná autentizace), která slouží k ochraně komunikace proti útokům typu MITM (Man In the Middle). Může implementovat i oboustrannou autentizaci, kdy klient poskytuje své autentizační údaje server. V praxi se tento přístup příliš nevyužívá. Pro ověřování identity se využívají veřejné klíče a certifikáty X.509, které jsou podepsány certifikační autoritou.
- *Důvěrnost (utajení)* – po navázání spojení a výměně konfiguračních parametrů je obsah přenášené komunikace šifrován.
- *Integrita dat* – TLS zajišťuje ochranu proti modifikaci, vložení či odebrání dat z přenášené komunikace pomocí tzv. kryptografického heše (např. MD4, SHA).

Samotný protokol TLS se skládá ze dvou dílčích protokolů, které jsou součástí zprávy TLS:

- *Handshake protocol* se používá při navazování spojení a zajišťuje autentizaci komunikujících stran, výběr kryptografických algoritmů, výměnu parametrů a sdílených klíčů.
- *Record protocol* zajišťuje bezpečný přenos mezi komunikujícími stranami.

[5] Viz `https://transparencyreport.google.com/https/overview?hl=en`

[6] Neformální popis popis standardu TLS je na `https://www.davidwong.fr/tls13/`.

7.1 Certi kacn autority

Certifikační autorita (CA) je v asymetrické kryptografii subjekt, který vydává digitální (elektronické) certifikáty a ověřuje jejich pravost. Certifikát je elektronický dokument (popsaný v notaci X.500) potvrzující příslušnost veřejného klíče k dané entitě (službě, serveru, osobě). Certifikační autorita na základě požadavku vygeneruje pro danou entitu dvojici veřejný a soukromý klíč a veřejný klíč s identifikací žadatele vloží do dokumentu zvaného certifikát. Struktura certifikátu je definována standardem X.509 a obsahuje následující informace [3]:

- jméno žadatele (subject)
- veřejný klíč žadatele (subject public key)
- informace o certifikátu (verze, sériové číslo, doba platnosti)
- informace o vydavateli (issuer)
- podpis certifikátu (certificate signature), algoritmus a parametry podpisu

Pro validaci certifikátu se využívá podpisu certifikační autority, která pomocí svého soukromého klíče podepíše vydaný certifikát. Ověření probíhá pomocí veřejného klíče CA. Jak můžeme ale veřejnému klíči CA důvěřovat? Tento klíč opět musí být podepsán, tzn. musí existovat certifikát, který ověřuje pravost veřejného klíče dané CA. Toto ověření provádí CA vyšší úrovně a vytváří se struktura důvěryhodnosti veřejných klíčů PKI (Public Key Infrastructure).

Pro ověřování certifikátů nejvyšší úrovně se používají národní a nadnárodní certifikační autority, jejichž veřejné klíče jsou obvykle umístěny přímo ve webových prohlížečích a dalších programech, čímž usnadní uživatelům ověřování důvěryhodnosti webových certifikátů či jiných elektronicky podepsaných dokumentů.

7.2 Transparentnost certi katu

Aby bylo možné ověřit, že CA vydávají certifikáty poctivě, byl experimentálně zaveden mechanismus transparentnosti certifikátů. Zapojené CA veřejně ohlašují každý vystavený certifikát (případně pre-certifikát). Vydané certifikáty se shromažďují v logovacím souboru `certificate transparency log`, kde může kdokoliv monitorovat nově přidávané certifikáty a auditovat přítomnost certifikátu nalezených při prohlížení webu.

8 System DNS (Domain Name System)

Systém DNS slouží jako podpora internetové komunikace, kdy ukládá informace důležité ke správnému fungování počítačových sítí a služeb. Jednou z nejvíce využívaných služeb je překlad doménového jména na IP adresu a naopak, tzn. vyhledávání záznamů typu A, AAAA a PTR [16, 21].

Mezi další důležité záznamy patří záznam `MX`, který obsahuje adresu poštovního serveru pro danou doménu. Záznamy `MX` jsou nezbytné pro správné fungování systému elektronické pošty nad SMTP. Pro vyhledávání serverů dalších služeb (např. SIP, XMPP a další) slouží záznamy typu `SRV`, které lze v současnosti využít pro služby SIP a XMPP [8].

Některé záznamy během vývoje internetu rozšířili svou funkci. Například záznamy typu `TXT` byly původně používány pro uložení textových informací o dané doméně. V současné době se převážně využívají pro zajištění bezpečnosti softwarových služeb či přenosu elektronické pošty, kde záznam `TXT` například obsahuje veřejný klíč pro službu DKIM (DomainKeys Identified Mail) [4]).

Služba DNS pracuje na principu klient–server, kde klient (zvaný též resolver) posílá dotaz pro vyhledání informací v distribuované globálním systému DNS. Dotaz přijde na server DNS, který se snaží najít odpověď bud v lokální databázi (zónovém souboru) či paměti cache. Pokud odpověď nenajde lokálně, může se dotazovat dalších serverů (v případě rekurzivního dotazu). Proces vyhledání informací v systému DNS se nazývá *rezoluce*.

8.1 Kon gurace klienta DNS

Co je klient DNS? Možná zvláštní otázka, nicméně definice klienta DNS není zcela jednoznačná. Za klientské programy lze považovat aplikace typu `nslookup`, `host` či `dig`, což jsou programy, které komunikují se servery DNS. Všechny tyto programy pro svou činnost využívají soubor systémových rutin zvaný *resolver*, který je součástí operačního systému a implementuje zasílání dotazů na server DNS a zpracování odpovědí. Z uživatelského pohledu lze tedy za klienty DNS považovat aplikace typu `nslookup`, z implementačního pohledu je klientem DNS *resolver*. Protože však aplikační programy typu `nslookup` umožňují uživateli přistupovat k systému DNS, budeme je také nazývat klienty DNS.

Pro konfiguraci klienta DNS se využívají následující konfigurační soubory: `/etc/hosts`, `/etc/resolv.conf` a `/etc/hosts.conf`.

- `/etc/hosts` – Tento soubor obsahuje statické mapování IP adres a doménových jmen. Vytváří se manuálně a využívá se například jako záloha v situaci, kdy není server DNS dostupný. Soubor `/etc/hosts` obvykle obsahuje jen základní mapování lokální adresy `localhost` na IP adresu. Do souboru můžeme vložit i další doménová jména a odpovídající IP adresy, viz následující ukázka. Formát souboru lze najít v manuálových stránkách `man hosts`.

```
::1         localhost localhost.my.domain
127.0.0.1   localhost localhost.my.domain
10.0.0.3    myfriend.my.domain myfriend
```

- `/etc/resolv.conf` – Jedná se o konfigurační soubor resolveru DNS. Resolver DNS vytváří dynamické mapování (nejen) IP adres a doménových jmen zasíláním dotazů na server DNS. Konfigurační soubor obsahuje například název implicitní domény (parametr `search`). Ta se doplňuje za doménová jména, která nejsou plně kvalifikovaná, např. doménové jméno `merlin` se při dotazování DNS automaticky rozšíří o implicitní doménu na `merlin.fit.vutbr.cz`. Dále obsahuje konfigurační soubor `/etc/resolv.conf` seznam serverů DNS (parametr `nameserver`), na které lokální počítač posílá dotazy DNS. Popis všech dostupných parametrů lze najít na manuálových stránkách `man resolv.conf`. Konfigurace souboru je buď manuální, tj. administrátor ručně vloží IP adresy lokálních serverů DNS, implicitní doménu apod. do tohoto souboru, nebo se konfigurace vytvoří z DHCP. Pokud je nastavena možnost konfigurace DHCP, pak se manuálně vložené data přepíšou hodnotami z DHCP. Příklad konfigurace lokálního klienta DNS na FIT:

```
# Generated by resolvconf
search fit.vutbr.cz             # implicitní doména
nameserver 147.229.9.43         # seznam serverů DNS
nameserver 147.229.8.43
```

- `/etc/host.conf` – U rezoluce doménových jmen je potřeba stanovit, zda má přednost statický překlad uložený v souboru `/etc/hosts` nebo dynamické vyhledávání systému DNS přes `resolver`. Pořadí rezoluce definuje konfigurační soubor `/etc/host.conf`, který je automaticky generován z konfiguračního souboru `nsswitch.conf`, viz `man nsswitch.conf`. Soubor `/etc/host.conf` udává pořadí zdrojů překladu DNS. Standardně se nejprve využívá statické mapování ze souboru `/etc/hosts` a teprve když se dané doménové jméno nenajde, dotazuje se lokální resolver serveru DNS, viz následující ukázka konfigurace:

  ```
  # Auto-generated from nsswitch.conf
  hosts
  dns
  ```

8.2 Konfigurace serveru DNS

Konfiguraci serveru DNS tvoří jednak konfigurace démona DNS (např. program `named`, což je implementace serveru Bind) a dále zónové soubory, které obsahují záznamy DNS pro domény, které server spravuje. Konfigurace serveru DNS je uložena v konfiguračním souboru `/etc/named.conf`, zónové soubory se ukládají do podadresáře `/var/named/`.

8.2.1 Konfigurační soubor `named.conf`

Soubor `named.conf` je základní konfigurační soubor aplikace `named`, viz `man named.conf`. Tento soubor specifikuje, jak má lokální server DNS zpracovat přijatý dotaz DNS. Server se nejprve pokusí najít hledané doménu v sekcích `zone` v konfiguračním souboru `/etc/named.conf`. Pokud ji najde, začne prohledávat příslušný zónový soubor a v něm všechny platné záznamy dané domény.

Pokud se jedná o lokálně spravovanou doménu, odpověď se hledá v lokálních zónových souborech umístěných na daném serveru v adresáři `/var/named/`. Pokud server nenajde hledanou doménu v žádné sekci `zone` konfiguračního souboru `/etc/named.conf`, znamená to, že doména není lokální. V takovém případě se pokusí server (v případě rekurzivního vyhledávání) poslat dotaz na kořenový server DNS ležící ve speciální zóně ".", která má typ `hint`. Tato zóna je obvykle součástí konfigurace serveru DNS.

Zónový soubor kořenové domény mívá standardizované jméno `named.root` (někdy také `root.zone`) a obsahuje záznamy typu A a AAAA všech třinácti kořenových serverů DNS. Pokud zóna "."v konfiguračním souboru chybí a hledaná doména není lokální, vrátí server negativní odpověď.

Níže uvedený příklad obsahuje dvě lokální domény se jmény `moje-domena.cz` a `10.10.10.in-addr.arpa`. U každé domény je odkaz na příslušný zónový soubor, v němž jsou uloženy všechny záznamy dané domény. Zároveň níže uvedená konfigurace obsahuje odkaz na kořenové servery (zóna typu `hint`).

```
...
zone "." IN {                       # zóna "." typu hint odkazující na kořenové servery DNS
  type hint;
  file "/etc/named/named.root";     # seznam kořenových serverů
};                                  # viz též https://www.internic.net/domain/named.root

zone "moje-domena.cz" IN {          # lokální doména se jménem "moje-domena.cz"
  type master;
  file "/var/named/moje-domena.cz"; # název zónového souboru obsahující záznamy dané domény
  notify no;
};
```

```
zone "10.10.10.in-addr.arpa" IN {    # lokální reverzní doména se jménem "10.10.10.in-addr.arpa"
  type master;
  file "/etc/named/10.10.10";        # název zónového souboru obsahující reverzní doménu
  notify no;
};
```

Soubor `named.root`, který obsahuje záznamy A a AAAA kořenových serverů, bývá součástí instalace serveru DNS. Po instalaci dojde k aktualizaci tohoto seznamu podle veřejného seznamu spravovaného službou InterNIC. Soubor `named.root` lze také získat programem dig:

```
dig +norec NS .  > db.root
```

8.2.2 Vytvoření vlastní domény

Pro každou nově vytvořenou doménu je potřeba přidat do konfiguračního souboru `/etc/named.conf` sekci `zone` se jménem nové domény a typem `master`. Dále je potřeba pro tuto doménu vytvořit zónový soubor, který bude obsahovat všechny potřebné záznamy dané domény. Povinné záznamy jsou SOA a NS, volitelné pak A, AAAA, MX, CNAME a další. Pokud přidáváme zónový soubor pro svou doménu, je vhodné vytvořit i reverzní zónový soubor pro zpětný překlad IP adres na doménová jména. Tento reverzní zónový soubor kromě povinných záznamů SOA a NS obsahuje jen záznamy typu PTR. Počet záznamů PTR by měl odpovídat počtu záznamů A a AAAA v přímém zónovém souboru, tj. aby pro každou IP adresu bylo možné najít příslušné doménové jméno.

Příklad konfigurace pro nově vytvářenou doménu `isa.cz` v `/etc/named.conf`:

```
zone "isa.cz" {
  type master;
  file "/var/named/isa.cz";
};
```

Jednotlivé záznamy v doméně `isa.cz` budou uloženy v souboru `/var/named/isa.cz`. Název souboru může být libovolný (např. `isa.cz`, `cz.isa`, `db.isa.cz`, `isa.cz.zone`), záleží na zvyklostech administrátora DNS. Název zónového souboru nemá vliv na jméno vytvářené domény. Podstatné je uvést jméno domény za klíčové slovo `zone`.

Formát zónového souboru obvykle obsahuje na začátku implicitní dobu životnosti dané zóny (klíčové slovo `$TTL`) a dále seznam jednotlivých záznamů pro danou doménu. Pro vytváření zónového souboru platí několik pravidel:

- Bílé znaky (prázdné řádky, odsazení) nejsou v konfiguraci důležité.
- Znak "@"má speciální význam - nahrazuje jméno domény, které je uvedeno v konfiguračním souboru za klíčovým slovem `zone`. Tj. v našem případě se každý výskyt znaku "@"v zónovém souboru nahradí řetězce "isa.cz".
- Základní formát záznamu DNS je následující

```
        <domain-name> <TTL> <CLASS> <TYPE> <rdata>
  např.   isa.cz.              IN      NS   ns.isa.cz.
```

  - `<domain-name>` obsahuje doménové jméno, které se vyhledává. Toto jméno musí být tzv. plně kvalifikované (FQDN, Fully Qualified Domain Name), tedy musí obsahovat cestu v doménovém stromě až ke kořeni (včetně ukončení tečkou). Pokud doménové jméno není ukončení tečkou, automaticky se za něj doplní doména uvedená v názvu zóny. Pokud bychom

například napsali pouze "isa.cz", pak DNS server doplní jména na "isa.cz.isa.cz.", což je asi něco jiného, než jsme požadovali. Naopak, pokud zapíše pouze název server "ns"(zkrácený zápis), pak server doplní název domény na "ns.isa.cz.", což je v pořádku.

Pro záznamy typu `SOA, NS, MX`, které popisují vlastnosti domény (tj. celého podstromu ve jmenném prostoru DNS), obsahuje toto políčko jméno dané domény, např. "isa.cz.". Pro záznamy záznamy typu `A, AAAA, CNAME`, které popisují vlastnosti konkrétního koncového uzlu jmenného prostoru DNS (například IP adresu, alias), obsahuje toto políčko doménové jméno (hostname) daného zařízení, např. "ns.isa.cz.", "pc1.isa.cz.", "www.isa.cz."apod.

- `<TTL>` udává dobu platnosti záznamu. Pokud tato hodnota není v popisu záznamu uvedena (jako v našem případě), vezme se implicitní hodnota uvedená na začátku zónového souboru za klíčovým slovem `$TTL`.
- `<CLASS>` popisuje třídu záznamů DNS. Historicky existovalo více různých tříd záznamů DNS, v současné době se používá pouze jediná třída typu `IN` (Internet). Tuto hodnotu musí záznam DNS vždy obsahovat.
- `<TYPE>` obsahuje typ záznamu DNS, například `SOA, NS, A, MX, CNAME, PTR` apod. Typ záznamu určuje, jaký bude formát položky `rdata` na pravé straně záznamu.
- `<rdata>` obsahuje vlastní data uložená v DNS pro dané doménové jméno. Obsah dat se liší podle typu záznamu. Například záznam typu NS očekává na pravé straně doménové jméno serveru DNS pro danou doménu. Naopak záznam typu SOA obsahuje základní nastavení dané zóny (jméno primárního serveru DNS, e-mailovou adresu správce, hodnoty TTL pro aktualizaci záznamů apod.).

- Znak ";"se používá jako oddělovač komentářů.

8.2.3 Příklad záznamů DNS v přímém zónovém souboru

Jak již bylo řečeno, každý zónový soubor obsahuje právě jeden záznam SOA s údaji o dané zóně a minimálně jeden záznam typu NS s adresou autoritativního serveru DNS pro danou doménu. Kromě těchto záznamů obvykle obsahuje přímý zónový soubor záznamy typu A (překlad doménových jmen na IPv4 adresy), záznamy typu AAAA (překlad doménových jmen na adresy IPv6), záznamy typu MX (odkazujících na poštovní server pro danou doménu), záznamy typu CNAME (aliasy koncových zařízení) a další.

- SOA – základní informace o zóně

```
@ IN  SOA ns.isa.cz. admin.isa.cz. ( ; primární server DNS, e-mail na správce
          2023092501   ; seriové číslo - datum editace, pořadí 01
          1w  ; refresh: interval obnovy pro sekundární servery (1 týden)
          1d  ; retry: po jaké době se pokusí sekundární server kontaktovat
              ; primární server, pokud se předchozí spojení nezdařilo (1 den)
          1w  ; expire: po jaké době je záznam neplatný, pokud se nepodařila
              ; aktualizace
          1h  ; negative cache TTL: TTL pro chybové odpovědi (např. NXDOMAIN)
  )
```

- NS – autoritativní servery DNS

```
@       IN NS ns1.isa.cz.  ; zápis využívající zástupný symbol "@"
isa.cz. IN NS ns1.isa.cz.  ; ekvivalentní zápis s plným doménovým jménem
@       IN NS ns1          ; zkrácený  zápis
```

- MX – poštovní server pro danou doménu (s prioritou)

```
isa.cz. IN MX 10 mail.isa.cz. ; primární poštovní server
isa.cz. IN MX 20 mail2.isa.cz.; sekundární poštovní server
isa.cz. IN MX 20 mail2        ; zkrácený  zápis
@       IN MX 20 mail2        ; zkrácený  zápis
```

- A – překlad IPv4 adres na doménová jména

```
ns.isa.cz. IN A 192.168.1.1  ; plně kvalifikované doménové jméno
ns         IN A 192.168.1.1  ; zkrácený zápis
pc1        IN A 192.168.1.2  ; IP adresa pro pc1.isa.cz.
pc2        IN A 192.168.1.3  ; IP adresa pro pc2.isa.cz.
```

- AAAA – překlad adres IPv6 pro doménová jména

```
pc3        IN AAAA fd12:1234::1  ; adresa IPv6 pro pc3.isa.cz.
```

- CNAME – alias počítače

```
www.isa.cz.  IN CNAME pc1.isa.cz. ; alias www pro PC1
www          IN CNAME pc1         ; zkrácený zápis
mail         IN CNAME pc1         ; alias pro server mail.isa.cz.
mail2        IN CNAME pc2         ; alias pro server mail2.isa.cz.
```

8.2.4 Reverzní zónový soubor

Reverzní zónový soubor obsahuje záznamy typu `PTR`, které překládají IP adresu na doménové jméno. IP adresa se v reverzním stromu nachází ve speciální podstromu `in-addr.arpa.` pro adresy IPv4 nebo v podstromu `ip6.arpa` pro IPv6. Záznamy typu `PTR` se vytvářejí pro každý záznam `A` či `AAAA` v přímém zónovém souboru. V našem příkladu máme čtyři doménová jména `ns.isa.cz, pc1.isa.cz, pc2.isa.cz` a `pc3.isa.cz`, pro která budeme vytvářet reverzní záznamy.

Reverzní záznamy jsou uloženy v jiném zónovém souboru než přímé. Jako přímého zónového souboru jsme použili jméno domény (např. "isa.cz"). Pro název reverzního zónového souboru použijeme prefix podsítě, do které počítače v naší doméně patří. Prefix se zapisuje v opačné pořadí bytů, tedy pro podsíť IPv4 `192.168.1.0` bude název reverzní zóny "1.168.192.in-addr.arpa". Tento název se objeví i v konfiguračním souboru `/etc/named.conf`, kde pro reverzní překlad vytvoříme novou zónu:

```
zone "1.168.192.in-addr.arpa" {
  type master;
  file "/var/named/192.168.1";
};
```

Reverzní zónový soubor jsme v tomto případě pojmenovali `192.168.1`, nicméně název není důležitý a závisí na administrátorovi DNS. Podobně jako přímý zónový soubor musí reverzní zónový soubor obsahovat dobu platnosti zóny `$TTL` a povinné záznamy SOA a NS. Dále obsahuje jen záznamy PTR.

Příklad reverzního zónového souboru pro doménu `isa.cz` je v následující ukázce:

```
$TTL 3h
@       IN SOA ns.isa.cz. admin.isa.cz. (            ; SOA záznam
              2023092501 ; serial number
              1w         ; refresh
              1d         ; retry
              1w         ; expire
              1h         ; negative cache TTL
        )
1.168.192.in-addr.arpa.   IN NS ns.isa.cz.  ; záznam NS
@                         IN NS ns.isa.cz.  ; zkrácený zápis

1.1.168.192.in-addr.arpa. IN PTR ns.isa.cz.  ; reverzní překlad pro ns.isa.cz.
1                         IN PTR ns.isa.cz.  ; zkrácený zápis
2                         IN PTR pc1.isa.cz. ; zkrácený zápis pro pc1.isa.cz.
3                         IN PTR pc2.isa.cz. ; zkrácený zápis pro pc2.isa.cz.
```

Při zápisu je potřeba si uvědomit, že na pravé straně musí být vždy plně kvalifikované doménové jméno. Pokud bychom na pravé straně uvedli například jenom "ns", tak po vyhodnocení dostaneme "ns.1.168.192.in-addr.arpa.", což asi nechceme. Podobně když zapomeneme tečku, tj. napíšeme pouze "ns.isa.cz", server vrátí hodnotu "ns.isa.cz.1.168.192.in-addr.arpa.".

Pro reverzní překlad IPv6 musíme vytvořit další zónový soubor, protože reverzní adresy IPv6 se nacházejí v jiném podstromu DNS (doméně) než adresy IPv4. Do konfiguračního souboru `/etc/named.conf` přidáme zónu "2.1.d.f.ip6.arpa.", viz následující ukázka:

```
zone "2.1.d.f.ip6.arpa" {     # odpovídá prefixu sítě fd12::
  type master;
  file "/var/named/fd12";
};
```

Tento reverzní zónový soubor pro zpětný překlad adres IPv6 na doménová jména bude opět obsahovat záznamy SOA a NS, dále pak záznamy PTR v doméně "2.1.d.f.ip6.arpa", viz následující ukázka:

```
$TTL 3h
@       IN SOA ns.isa.cz. admin.isa.cz. (            ; SOA záznam
              2023092501 ; serial number
              1w         ; refresh
              1d         ; retry
              1w         ; expire
              1h         ; negative cache TTL
        )
2.1.d.f.ip6.arpa.   IN NS ns.isa.cz.  ; záznam NS
@                   IN NS ns.isa.cz.  ; zkrácený zápis
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.3.2.1.2.1.d.f.ip6.arpa. IN PTR pc3.isa.cz.
                                      ; plný zápis pro adresu fd12:1234::1
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.3.2.1 IN PTR pc3.isa.cz.
                                      ; zkrácený zápis pro adresu fd12:1234::1
```

Nevýhodou reverzního zápisu adresy IPv6 je, že záznamy DNS neznají zkracování nulových bytů. Je tedy nutné uvést celou cestu v reverzním stromu DNS pro IPv6, kde každá úroveň obsahuje jednu hexadecimální číslici adresy IPv6 reprezentující čtyři bity. Adresa IPv6 tedy obsahuje 32 hexadecimálních

číslic, které je nutné v reverzním překladu rozepsat. Protože tento výpis je značně složitý a může dojít snad k chybě, používají se na vytváření reverzních adres specializované generátory[7].

8.2.5 Kontrola záznamů a restart serveru DNS

Po každé změně zónového souboru by se vždy mělo upravit i sériové číslo zóny tak, aby bylo vždy větší než předchozí hodnota. Standardně se do sériového čísla vkládá datum a pořadí změny, tj. má formát `yyyymmddxx`, kde `xx` je pořadí změny v daném dni.

Pokud spouštíme službu DNS poprvé, lze použít příkaz:

```
systemctl start named.service
```

Pokud restartujeme běžící službu DNS, lze použít příkaz:

```
systemctl restart named.service
```

Následujícím příkazem ověříme, zda služba běží či nikoliv:

```
systemctl status named
```

V případě syntaktické či sémantické chyby v zónových souborech či konfiguraci serveru DNS, proces nahlásí chybu a ukončí se. Z popisu chyby se dát obvykle snadno vyčíst, na kterém řádku konfigurace nastala chyba.

[7] Viz například `https://www.whatsmydns.net/reverse-dns-generator`.

9 Prenos systemovych udalosti sluzbou Syslog

Služba Syslog slouží pro generování a přenos systémových událostí síťových zařízení a koncových stanic na centrální úložiště (server). Systémové událostí jsou generovány operačním systémem či běžícími aplikace. Tyto události se lokálně ukládají do logovacích souborů. Služba Syslog posílá vygenerované události na server.

Architekturu služby Syslog tvoří protokol Syslog [7], který popisuje formát přenosu zpráv, dále Syslog klient (`originator`), které vytvořené události posílá na server, a Syslog server (`collector`), který přijímá poslané události protokolem Syslog od klientů a centrálně je zpracovává.

Každá vygenerovaná událost se ukládá jako záznam do logovacího souboru buď operačního systému (např. `/var/log/messages`) nebo do logovacího souboru konkrétní aplikace (např. `/var/log/maillog` pro poštovní server, `/var/log/ipfw` pro firewall, `/var/log/apache/access.log` pro HTTP server).

Logovací záznam události zpravidla obsahuje časovou značku, typ zařízení, které událost vygenerovalo (facility), závažnost události (severity) a slovní popis události. V některých logovacích souborech se může objevit i doménové jméno či IP adresa stanice, která událost způsobila. Záznamy mohou být uloženy lokálně v souborovém systému (na unixových systémech v adresáři `/var/log`) nebo je můžeme přeposílat protokolem Syslog na vzdálený server.

Typ zařízení, které událost vygenerovalo, je definováno identifikátorem v záznamu události. Seznam zařízení, které definuje protokol Syslog, je v tabulce 3. Protože tento seznam vychází z původního

Code	Facility	Code	Facility
0	kernel messages	12	NTP subsystem
1	user-level messages	13	log audit
2	mail system	14	log alert
3	system daemons	15	clock daemon (note 2)
4	security/authorization messages	16	local use 0 (local0)
5	messages generated internally by syslogd	17	local use 1 (local1)
6	line printer subsystem	18	local use 2 (local2)
7	network news subsystem	19	local use 3 (local3)
8	UUCP subsystem	20	local use 4 (local4)
9	clock daemon	21	local use 5 (local5)
10	security/authorization messages	22	local use 6 (local6)
11	FTP daemon	23	local use 7 (local7)

Tabulka 3: Typy zařízení Syslog dle RFC 5424 [7]

návrhu protokolu BSD Syslog z roku 2001, obsahuje zařízení, která se dnes už příliš nepoužívají. Z tohoto důvodu se pro aplikace a další typy zdrojů událostí používá lokální označení s kódy 16-23.

Pro hodnocení závažnosti vzniklé události definuje Syslog osm úrovní od nejzávažnější typu Emergency (0) po ladící výpisy typu Debug (7). Pro správu sítí jsou důležité zejména události od úrovně Warning (4). Přehled definovaných úrovní závažnosti je v tabulce 4.

Code	Severity	Description
0	EMERGENCY	system is unusable
1	ALERT	action must be taken immediately
2	CRITICAL	critical conditions
3	ERROR	error conditions
4	WARNING	warning conditions
5	NOTICE	normal but significant condition
6	INFORMATIONAL	informational messages
7	DEBUG	debug-level messages

Tabulka 4: Úrovně závažnosti událostí Syslog dle RFC 5424 [7]

10 Monitorovan s te SNMP

Služba SNMP (Simple Network Monitoring Protocol) slouží k aktivnímu monitorování zařízení na síti. Protokol SNMP [2] slouží k přenosu monitorovacích informací mezi řídící stanicí NMS (Network Management Station) a monitorovaným zařízením (SNMP Agent). Protokol slouží nejen ke čtení provozních parametrů (hodnot objektů MIB) připojených stanic (využití paměti, počty přenesených paketů, stav síťových rozhraní, zatížení CPU apod.), ale dá se použít i k nastavení některých těchto parametrů. Protokol pracuje nad UDP na portech 161 (vyzývání) a 162 (asynchronní zprávy, traps).

Řídicí stanice NMS posílá zprávy SNMP na jednotlivá síťová zařízení s cílem získat hodnoty sledovaných objektů systému. Standardní způsob komunikace pracuje na principu vyzývání (polling), kdy řídicí stanice NMS periodicky posílá zprávy typu dotaz na sledovaná zařízení. Požadované hodnoty (objekty MIB) jsou identifikovány pomocí tzv. OID (Object Identifier), které jsou standardizovány v databázi MIB (Management Information Base).

Příkladem objektu je `sysDescr` (System Description) s OID `1.3.6.1.2.1.1.6.0`. Standardizované objekty jsou uloženy v databázi MIB-2, která je definována standardem RFC 1213 [13].

Pro vyhledávání objektů SNMP lze využít následující portály:

- Global OID Reference Database: `http://oidref.com/`
- Object Identifier (OID) Repository: `http://www.oid-info.com/`

10.1 Nastroj pro monitorovan zar zen SNMP

Pro monitorování zařízení SNMP můžeme využít nástroj `snmpwalk`, viz `man snmpwalk`, který slouží k získání objektů z daného zařízení protokolem SNMP. Program `snmpwalk` vrátí buď požadovaný objekt MIB zadaný identifikátorem OID nebo množinu objektů MIB, které patří do zadaného podstromu MIB.

Formát a příklad příkazu `snmpwalk`:

```
snmpwalk -v <protocol> -c <community string> <host> <object>

snmpwalk -v2c -c public isa.fit.vutbr.cz sysDescr         ; vyhledá objekt System Description
snmpwalk -v2c -c public isa.fit.vutbr.cz 1.3.6.1.2.1.1.1  ; ekvivalentní zápis pomocí OID

snmpwalk -v2c -c public isa.fit.vutbr.cz system           ; vyhledá objekty podstromu System
snmpwalk -v2c -c public isa.fit.vutbr.cz 1.3.6.1.2.1.1    ; ekvivalentní zápis pomocí OID
```

10.2 Kon gurace agentu SNMP

Zařízení v síti SNMP jsou rozdělena do logických celků, komunit, identifikovaných textovým řetězcem. Komunita slouží k autentizaci zařízení v síti a definuje přístup k informacím MIB.

V konfiguraci monitorovaného zařízení (agenta SNMP) můžeme definovat, kdo může přistupovat k jakým objektů SNMP. Můžeme definovat způsob přístupu k objektům SNMP (pro čtení: `ro`, pro čtení a zápis: `rw`), můžeme omezit přístup na určité zdrojové adresy, či můžeme definovat část podstromu MIB, do kterého má daný uživatel přístup. Příklad nastavení:

```
rocommunity public 10.0.0.0/24
rocommunity public 192.168.0.0/24 1.3.6.1.2.1.1
rwcommunity public localhost
```

Výše uvedená konfigurace umožňuje zařízením v síti 10.0.0.0/24 přístup k objektům MIB komunity `public` pouze pro čtení. Stanice v síti 192.168.0.0/24 mají přístup pro čtení omezen pouze na základní systémové informace uložené v podstromu MIB-2 `system` (OID 1.3.6.1.2.1.1). (podstrom system). Pro lokálního uživatele budou objekty MIB přístupné pro čtení i pro zápis.

11 Monitorovan toku NetFlow

Služba NetFlow byla navržena pro monitorování toků v síti. Její architekturu tvoří sonda NetFlow (exportér), která monitoruje síťový provoz, analyzuje jednotlivé pakety na úrovni L3 a L4, a vytváří tzv. záznamy (flow records) o probíhajícím provozu. Záznamy jsou uložený v lokální paměti sondy, který je po skončení toku přepošle protokolem NetFlow na tzv. kolektor. Kolektor ukládá záznamy z různých sond do centrálního úložiště, kde je můžeme vyhledávat, analyzovat, identifikovat potenciálně nebezpečný provoz apod.

Příklad záznamu o toku:

```
Date flow start   Duration Proto Src IP Addr:Port      Dst IP Addr:Port   Flags Tos Packets Bytes Flows
2005-08-30 06:53:53 63.545 TC  113.138.32.152:15225 ->  222.33.70.124:3575.AP.SF 0    62     3512   1
2005-08-30 06:53:53 63.545 TC  222.33.70.124:3575   ->  113.138.32.152:25 .AP.SF 0    58     3300   1
```

Tento záznam obsahuje dva toky TCP, z nichž první tvoří 62 paketů o velikosti 3.512 B, které byly poslány z adresy 113.138.32.152 a portu 15225 an adresu 222.33.70.124, port 3575. Délka přenosu trvala 63,545 sekund.

11.1 Program `nfdump`

Program `nfdump` slouží jako kolektor systému NetFlow, viz `man nfdump`. Program také umožňuje zobrazovat uložené záznamy, vyhledávat v nich, provádět agregaci, filtrování a další.

```
Format:   nfdump [options] [filter]
```

Příklady použití:

```
nfdump -r nfcapd.202004221040  # vypíše všechny záznamy o tocích z daného souboru

nfdump -r nfcapd.202004221040 'host 147.229.5.2'  # vypíše všechny toky dané stanice

nfdump -r nfcapd1 -c 100 'proto tcp and (src ip 172.16.17.18 or dst ip 172.16.17.19)'
         # vybere ze souboru 100 záznamů, které vyhovují zadanému filtru

nfdump -r nfcapd1 -s record/bytes -n 10
         # vygeneruje statistiku Top 10 záznamů s největším počtem přenesených bytů

nfdump -r nfcapd1 -s srcip/packets -n 10
         # vygeneruje statistiku Top 10 IP adres s největším počtem odeslaných paketů

nfdump -r nfcapd1 -A srcip,dstport
         # vypíše agregované toky podle zdrojové IP adresy a cílového portu

nfdump -r nfcapd1 -A srcip 'host 147.229.5.2'
         # vypíše agregované toky podle zdrojové IP adresy pro daný filtr

nfdump -r nfcapd1 -t 2020/04/22.12:19:00-2020/04/22.12:20:00 'port 53'
         # vyhledá všechny záznamy s komunikací DNS v daném časovém okně
```

12 Signalizacn protokol SIP

Session Initiation Protocol (SIP, RFC 3261 [20]) je signalizační protokol pro IP telefonie, který provádí následující operace:

- registrace uživatele VoIP na serveru SIP
- navázání hovoru, výměna parametrů spojení
- směrování hovorů
- ukončení či zrušení hovoru

Protokol SIP neprovádí správu relací po jejich navázání, nezajišťuje kvalitu přenosu, neslouží k přenosu hlasových dat.

Základními prvky IP telefonie jsou *server UAS* (User Agent Server), který provádí registraci klientů, navazování a směrování hovorů, lokalizaci klienta účtování apod., a *uživatelský agent UAC* (User Agent Client), který zprostředkovává uživateli volání (vytáčení spojení, příjem telefonátu).

Služba SIP používá pro adresování speciální URI (Uniform Resource Identifier) ve tvaru `sip:user@domain`, kde `user` je uživatelské jméno, které přidělí poskytovatel uživateli, a `domain` je VoIP doména poskytovatele. Adresa uživatele je součástí metody `INVITE`, která slouží k vytvoření hovoru, nebo se využívá při registraci telefonu (metoda `REGISTER`).

Směrovací informace se během navazování spojení ukládají v hlavičce SIP paketu `Via:`, `Route` či `Record-Route`.

12.1 Pr klad signalizace SIP

```
INVITE sip:541141118@cesnet.cz SIP/2.0    # navázání spojení s uživatelem 541141118 v doméně cesnet.cz
Call-ID: D40CA785-2EEE-4801-9B04-349632F56CDC@147.229.14.146   # jednoznačný identifikátor hovoru
CSeq: 2 INVITE                                                 # sekvenční číslo zprávy, název metody
From: "Petr Matousek"<sip:matousp@cesnet.cz>;tag=2007034328740 # identifikace volajícího (SIP URI)
To: <sip:541151118@cesnet.cz>                                  # identifikace volaného (SIP URI)
Contact: <sip:matousp@147.229.14.146:59183;ob>                 # identifikace volaného (Device URI)

SIP/2.0 100 Trying -- your call is important to us             # odpověd' na metodu INVITE
Call-ID: D40CA785-2EEE-4801-9B04-349632F56CDC@147.229.14.146   # jednoznačný identifikátor hovoru
CSeq: 2 INVITE                                                 # odpověd' na metodu INVITE
From: "Petr Matousek"<sip:matousp@cesnet.cz>;tag=2007034328740 # zopakování hlaviček z metody INVITE
To: <sip:541151118@cesnet.cz>
```

12.2 Zakladn metody protokolu SIP

- REGISTER – žádost o registraci
- INVITE, ACK – vytváření spojení
- CANCEL – zrušení vytvářeného spojení
- BYE – ukončení spojení
- OPTIONS – zjišťování možností přenosu
- PRACK – provizorní ACK

12.3 Dals metody protokolu SIP

- INFO (RFC 6086) – přenos aplikačních informací
- MESSAGE (RFC 3428) – textové zprávy IM
- SUBSCRIBE, NOTIFY (RFC 3265) – zasílání upozornění na události
- PUBLISH (RFC 3903) – zveřejnění stavu událostí (prezence)

12.4 Polozky hlavicky SIP

- **Via:** obsahuje směrovací informace. Každý server, přes který zpráva `INVITE` prochází, přidá do hlavičky jeden řádek `Via:` s IP adresou a portem, na kterém zpracoval paket. Zaznamenaná posloupnost uzlů, přes které zpráva `INVITE` procházela, se pak využije pro směrování odpovědi.

  ```
  via: SIP/2.0/UDP 10.11.12.51:5062;rport;branch=z9hG4bk2079031997
  ```

- **From:** obsahuje odesílatele SIP zprávy ve formátu SIP URI. Nepovinně může obsahovat i zobrazitelné jméno (display-name), které uživatel vložil do konfigurace klienta.

  ```
  From: "user08" <sip:user08@iss.cz>;tag=283687066
  ```

- **To:** obsahuje adresu příjemce zprávy.

  ```
  To: <sip:1006@iss.cz>
  ```

- **Allow** obsahuje seznam metod SIP, které podporuje daný klient.

  ```
  Allow: INVITE, INFO, PRACK, ACK, BYE, CANCEL, OPTIONS, NOTIFY, REGISTER
  ```

12.5 Protokol SDP (Session Description Protocol)

Protokol SDP [9] se používá pro zaslání informací potřebných k vytvoření kanálu pro přenos hlasu (případně obrazu). Jde opět o signalizační protokol, který domlouvá parametry spojení, jako je typ média (video, audio, atd.), transportní protokol (RTP/UDP/IP, H.320, atd.), typ kodeku, IP adresa a port pro příjem hlasu (obrazu). Zprávy SDP se přenáší ve zprávách typu `INVITE` a `OK`.

```
v=0                                          # verze protokolu SDP
o=- 3434794301 3434794301 IN IP4 147.229.14.146 # původce spojení (username, IP adresa)
s=Sjphone                                    # jméno spojení: jméno softwaru
c=IN IP4 147.229.14.146                      # spojení: IP adresa pro příjem dat
t=0 0 # time (start + end)
a=direction:passive
m=audio 49152 RTP/AVP 3 97 98 8 0_101        # audio stream: číslo portu, nabízené kodeky
a=rtpmap:3 GSM/8000
a=rtpmap:97 iLBC/8000
a=rtpmap:98 iLBC/8000
a=fmtp:98 mode=20
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-11,16
```

Reference

[1] R. Barnes, M. Thomson, A. Pironti, and A. Langley. Deprecating Secure Sockets Layer Version 3.0. RFC 7568 (Proposed Standard), June 2015. Updated by RFC 8996.

[2] J. Case, D. Harrington, R. Presuhn, and B. Wijnen. Message Processing and Dispatching for the Simple Network Management Protocol (SNMP). RFC 3412 (Internet Standard), December 2002. Updated by RFC 5590.

[3] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008. Updated by RFCs 6818, 8398, 8399.

[4] D. Crocker (Ed.), T. Hansen (Ed.), and M. Kucherawy (Ed.). DomainKeys Identified Mail (DKIM) Signatures. RFC 6376 (Internet Standard), September 2011. Updated by RFCs 8301, 8463, 8553, 8616.

[5] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Obsoleted by RFC 8200, updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.

[6] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Historic), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176, 7465, 7507, 7919.

[7] R. Gerhards. The Syslog Protocol. RFC 5424 (Proposed Standard), March 2009.

[8] A. Gulbrandsen, P. Vixie, and L. Esibov. A DNS RR for specifying the location of services (DNS SRV). RFC 2782 (Proposed Standard), February 2000. Updated by RFCs 6335, 8553.

[9] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566 (Proposed Standard), July 2006. Obsoleted by RFC 8866.

[10] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), February 2006. Updated by RFCs 5952, 6052, 7136, 7346, 7371, 8064.

[11] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. RFC 4193 (Proposed Standard), October 2005.

[12] S. Kawamura and M. Kawashima. A Recommendation for IPv6 Address Text Representation. RFC 5952 (Proposed Standard), August 2010.

[13] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. RFC 1213 (Internet Standard), March 1991. Updated by RFCs 2011, 2012, 2013.

[14] D. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard), March 1992. Obsoleted by RFC 5905.

[15] D. Mills, J. Martin (Ed.), J. Burbank, and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905 (Proposed Standard), June 2010. Updated by RFCs 7822, 8573, 9109.

[16] P. Mockapetris. Domain names - implementation and specification. RFC 1035 (Internet Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604, 7766, 8482, 8490, 8767.

[17] T. Mrugalski, M. Siodelski, B. Volz, A. Yourtchenko, M. Richardson, S. Jiang, T. Lemon, and T. Winters. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 8415 (Proposed Standard), November 2018.

[18] J. Postel. Internet Protocol. RFC 791 (Internet Standard), September 1981. Updated by RFCs 1349, 2474, 6864.

[19] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard), August 2018.

[20] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878, 7462, 7463, 8217, 8591, 8760, 8898, 8996.

[21] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi. DNS Extensions to Support IP Version 6. RFC 3596 (Internet Standard), October 2003.

[22] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007. Updated by RFC 7527.

[23] T. Ylonen and C. Lonvick (Ed.). The Secure Shell (SSH) Transport Layer Protocol. RFC 4253 (Proposed Standard), January 2006. Updated by RFCs 6668, 8268, 8308, 8332, 8709, 8758, 9142.