

## L3 zodpovědnost

- směrování - adresa dalšího hopu a interface pro každý příchozí paket
- předávání - předání ze vstupního na výstupní interface
- fragmentace - pokud je zpráva moc velká
- notifikace - pokud něco selže (volitelné)

## MAC

- 48bit, <OID,NIC>

## Ethernet

- <pre ; start-of ; dst ; src ; len ; data ; seq\_n >

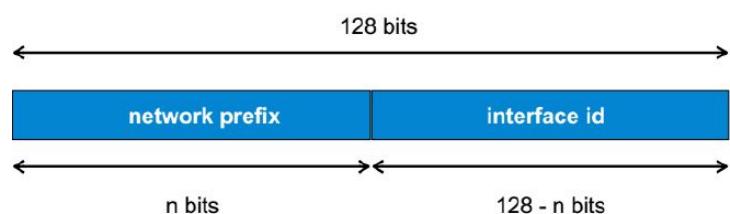
## IPv4 (32b)

- nespojovaná
- bez QoS (v základu)
- formát - < network ; host >
- classful - masky podle tříd
- classless - třídy ještě děleny na podsítě



## IPv6 (128b)

- fragmentují pouze end-pointy (ale když linka MTU nezvládne, musí rozložit a složit)
- větší default MTU - 1280b + MTU discovery
- nepodporuje broadcast - jen multicast
- hlavička 40B - <ver ; traff. class ; flow label ; len ; next\_head ; hop\_lim ; src ; dst >
- může následovat série extend hlaviček (každá další identifikována next\_head)
- obvykle /48 network prefix



## Global IPv6 anycast

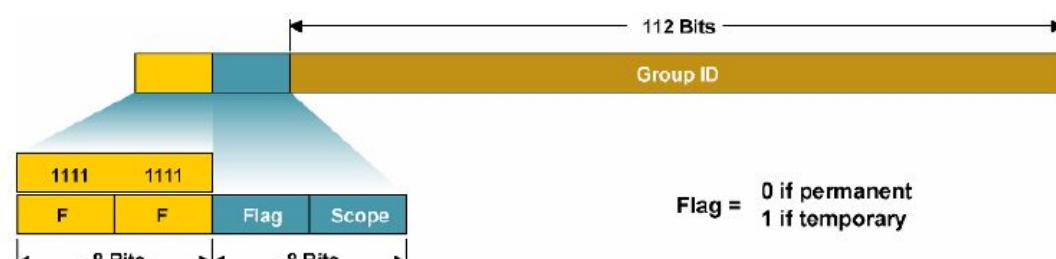
- SLAAC (stateless autoconf)
  - EUI-64:
    - Automatické nastavení IPv6 adresy - host id je odvozen z MAC a dvou doplnění dvou bytů vložených mezi OID a NIC -> <OID ; 0xFF ; 0xFE ; NIC> a invertováním universal / local bitu (7 bit) MAC adresy.
    - Privacy ext. - podobný princip, ale není přímo mac adresa.
    - DAD - duplicate address detection

## Link-local IPv6

- local prefix FE80::/10
- <FE80 ; random 54 bits ; EUI-64, nebo Privacy. >

## IPv6 multicast

- prefix FF00:: /8



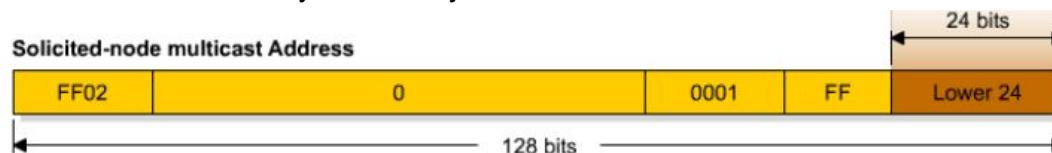
▪ **Multicast address** is frequently used

- Replaces broadcast
- Has prefix FF00::/8

	Meaning
FF02::1	All nodes
FF02::2	All routers
FF02::9	All RIP routers
FF02::1:FFXX:XXXX	Solicited-node

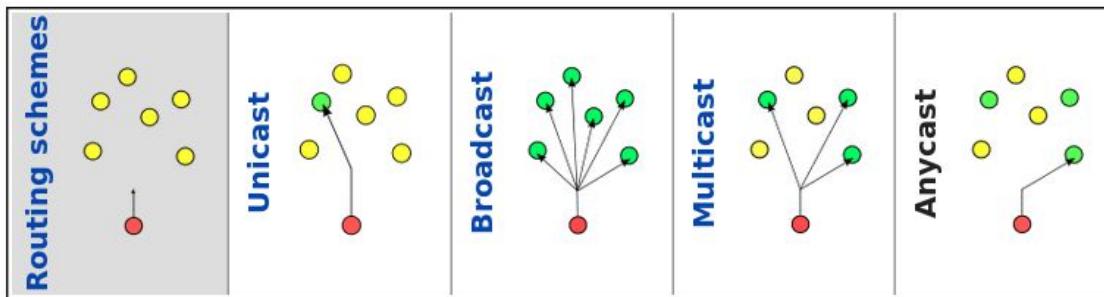
## Solicited-Node multicast address

- prefix FF02::1:FF:/104
- 24 bitů z unicast nebo anycast adresy



## Průchod paketů sítí

**Anycast** - více serverů, které mohou být cílem, zpráva se směruje na ten nejblíž



## ARP (Address Resolution Protocol)

- IPv4-to-MAC - získání linkové adresy síťového rozhraní protistrany ve stejné podsíti pomocí známé IP adresy
- funguje na principu request - reply, pokud nezná překlad, posílá request broadcast a ten, kdo ví, odpoví reply s MAC.
- ipv4 bez něj nefunguje

## NDP (Neighbor discovery protocol)

- ipv6 bez něj nefunguje
- běží na ICMPv6
- je zodpovědný za automatickou konfiguraci adres uzlů, objev zjišťování uzlů na lince, určování adresy linkové vrstvy jiných uzlů, hledání duplicit adres, hledání dostupných směrovačů a [DNS](#) serverů, zjišťování prefixů adres a udržování dosažitelnosti informace o cestách do jiných aktivních sousedních uzlů
- 5 typů zpráv:
  - Router solicitation / advert. - hledám router / nabízím router
  - Neighbor sol. / advert - hledám mac hosta, nebo ping / odpovídám mac
  - Redirect - router varuje odesílatele, že existuje lepší cesta

=====

## Questions

Describe IP fragmentation!

Explain operation of switch with CAM!

Describe IPv6 Extension headers!

What is the difference between unicast, multicast, broadcast and anycast? Which of them are present in IPv4 and IPv6?

What is routing and what is switching?

Describe ARP and ND L3-to-L2 resolution process!

Inform about various Internet layered models!

What is collision domain? Identify it on network diagram.

What is broadcast domain? Identify it on network diagram.

Explain IPv4 subnetting on example!

Compare hub and switch? What is modem?

How do you recognize IPv6 link-local address? What is the purpose of link-local addresses?

# Protokoly

## ICMP (Internet Control Message Protocol)

- IPv4 servisní protokol
- diagnóza sítě a hlášení chyb
- ping, traceroute

## ICMPv6

- vše jako ICMP, přidává zprávy pro router / neighbor solicitation / advertisement
- Základ pro NDP
- signalizační protokol - například Too big packet

## DHCP

- Discover, offer, request, ack

## SLAAC

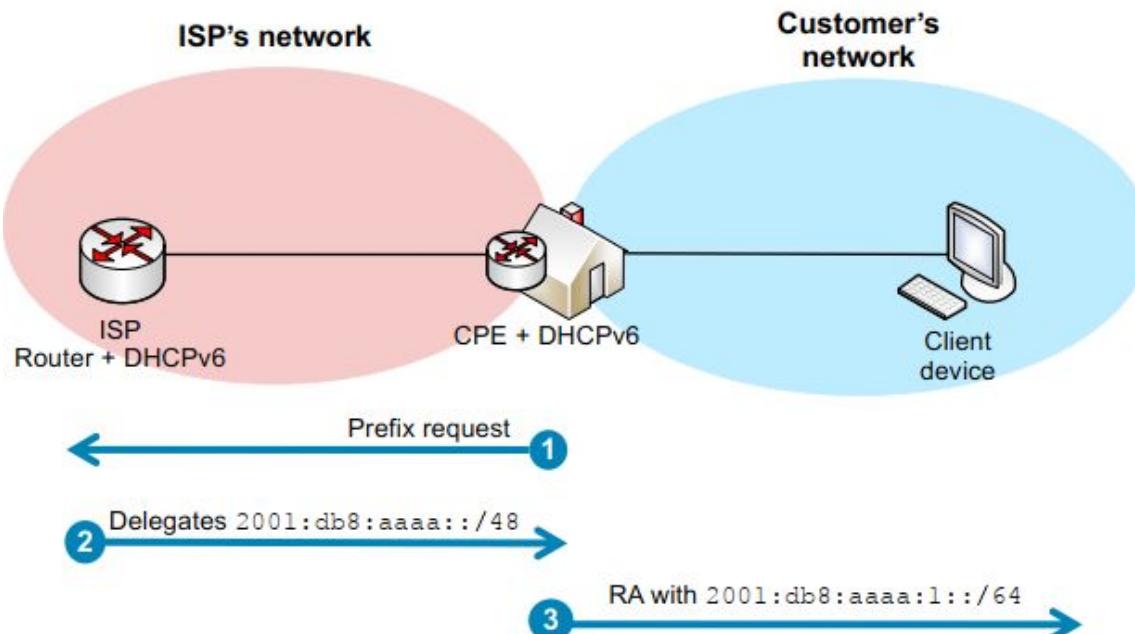
- viz předchozí strany.
- **SLAAC** (stateless autoconf)
  - EUI-64:
    - Automatické nastavení IPv6 adresy - host id je odvozen z MAC a dvou doplnění dvou bytů vložených mezi OID a NIC -> <OID ; 0xFF ; 0xFE ; NIC> a invertováním universal / local bitu (7 bit) MAC adresy.
    - Privacy ext. - podobný princip, ale není přímo mac adresa.
    - **DAD** - duplicate address detection

## DHCPv6

- stateless a stateful
- není povinný, některé zařízení můžou pracovat i bez něj
- Stateless:
  - adresy se nastaví SLAAC
  - problém detekovat DNS servery -> informace se zjišťují pomocí request / reply mezi sousedy
- Stateful
- **Prefix delegation**
  - ptám se providera na síťový prefix, se kterým potom adresuju v lokální síti.
  - (Protože není NAT a všechno jde napřímo)



## DHCPv6 Prefix delegation



## NAT

- nepotřebuje celý blok veřejných adres pro lok. síť, stačí jedna veřejná a celá podsíť se schová za ni.
- port forwarding
- musí nahlížet do TCP/UDP hlaviček -> zátěž

## DNS

- netřeba asi rozmazávat.
- primární, sekundární, cache

# Protokoly a chyby přenosu

## Chyby

### Bitové

- L2 - řešením obvykle redundance
- většinou chyby zpracování



### Paketové chyby

- L4 - chyby přenosu
  - znova poslat
- *K čemu může na úrovni paketů dojít?*
  - **Ztráta (Zpoždění)**
  - **Ztráta fragmentovaných dat**
  - **Duplikace**
  - **Vložení (Zpoždění)**
  - **Změna pořadí**

## Detekce chyb

### Sekvenční čísla

- inkrementováno pro každý fragment, při znovuzaslání se nemění
- po přetečení se nuluje - musí být dost velké, aby se vlivem zdržení v síti nedostal paket s přetečenou hodnotou na pozici předchozího (stačí 32bit)
- -> detekujeme duplicitu a pořadí

### Handshake

- na jedné straně vygenerován ISN a předán pouze druhé straně, která jej použije jako ESN, tyto hodnoty jsou pak základem pro sekvenční čísla.
- používají se SYN pakety, po dokončení komunikace FIN od ukončujícího a vyprázdnění ESN a ISN

### 2Way Handshake

- prakticky to stejné, ale vymění si ISN a ESN navzájem (každý má číslo protějšku).

### 3Way Handshake

- prakticky to stejné, ale ještě si je musí potvrdit (každý inkrementuje a pošle zpět)

## Detekce ztráty paketu

### Timeout

- na straně příjemce / odesílatele
- jakou zvolit hodnotu? Fixní může být průser, počítat tedy průměr round-trip time

### Negativní potvrzování

- pošlu seznam těch, co nedorazily
- nevýhoda - zahazování je obvyklé v případě zahlcení sítě, posílání opakovaných zpráv, že něco nedorazilo tomu moc nepomůže, navíc se může NACK taky ztratit.

# Korekce chyb

## Znovuzaslání

- Varianty:
  - **stop-and-wait**
    - něco nedorazilo, tak posílám a čekám, dokud nedorazí
    - implementováno jako klouzající okno
      - velikost okna 1 na obou stranách
    - malá efektivita
  - **go-back-n**
    - vrátím se tam, kde se to pokazilo a posílám znova
      - velikost okna N pro odesílání, 1 pro příjem
    - + jednoduchý
    - - plýtvá pásmem (posílá se i úspěšné)
    - Odesílá mi vždy ACK jen posledně dobré přijatého paketu
  - **selektivní (použito v TCP)**
    - pošlu jen to, co se ztratilo
    - + neplýtvá šírkou pásma
    - - složitější - nutnost řadit pakety - nezvládá více, než 1 chybu v okně
  - **SMART** (selektivní + go back)
    - + zvládne více chyb v okně
    - - stále se musí řadit

$$eff = \frac{1-p}{1-p+pw}$$

## Dodatky ke klouzavému oknu

- Jak na straně odesílatele, tak na straně příjemce
  - Každé může mít jinou délku
- Jak velké ho nastavit?
  - příliš malé zvětšuje režii protokolu
  - příliš velké zahlcení front middle-boxů
  - řešení = měnit ho podle chování sítě

# Transportní protokoly

## TCP

- spolehlivé doručení (znovuzasílání, řízení zahlcení)
- optimalizuje přenosovou rychlosť tak, aby nedocházelo k zahlcení
  - ztráta paketu == zahlceno -> posílám pomaleji
  - různé strategie změny rychlosti (pomalý náběh)
- Nevýhody
  - vše za ztraceným paketem musí počkat (HOL)
  - pokud se změní IP, musí se navázat znova spojení
  - velká režie na potvrzování

## UDP

- nezaručuje doručení
- tlačím data na plný plyn, žádné znovuzasílání, řešení pořadí etc.

## DCCP

- UDP s řízením zahlcení 48bit seq číslem. Pokud se jich ztratí moc, tak ubere plyn.
- 3-way handshake - musí se domluvit způsob řízení zahlcení

## SCTP

- spolehlivý
- kombinuje více zpráv do jedné
- místo byte stream message stream (zohledňuje jen hranice aplikačních dat)
- velká signalizační režie
- 4-way handshake - zabraňuje SYN-flood
- multihoming (nemusí se znova navazovat při změně ip)

- MTU discovery - ochrana před fragmentací
- multiplex TCP skrz více spojení
- menší signalizační režie než SCTP
- Vše co umí TCP + vlastní řízení zahlcení

### (Quick UDP Internet Conn.)

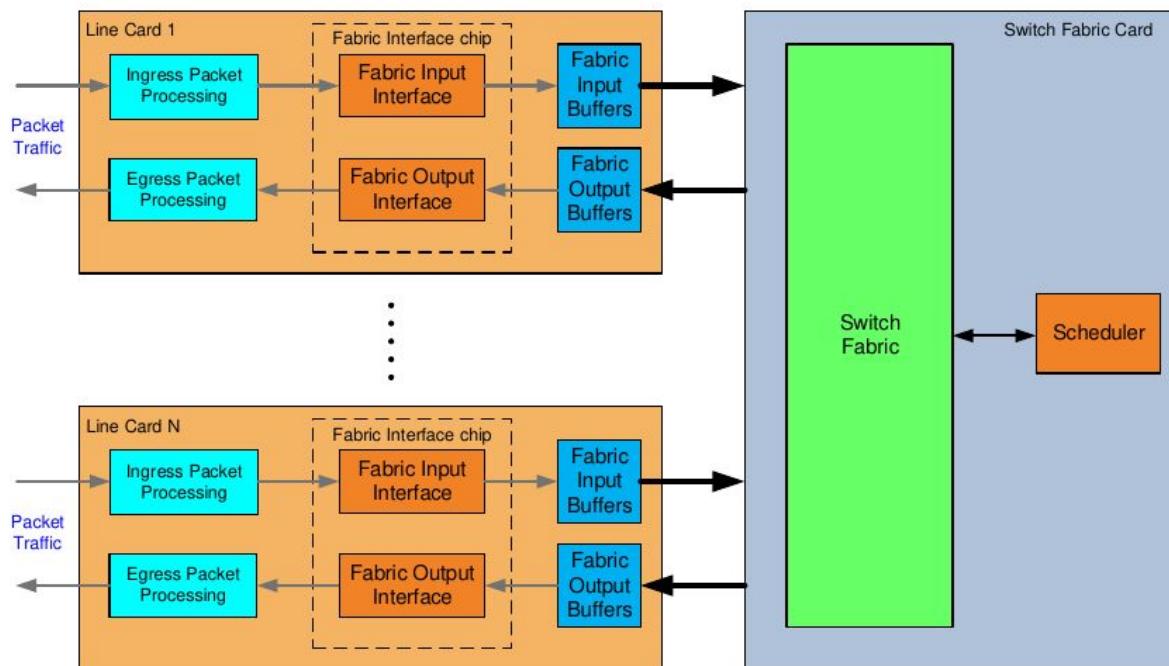
- Problematické navazování TCP spojení
- dokáže díky xor hashi obnovit ztracené data
- má kompresi -> zrychluje přenos
- má řízení zahlcení
- Vyvinul ho Google
- congestion control
- flow control
- řeší HOL
- TCP cubic
- SMART znovuodeslání

## Otzázkы

- Jmenuj a vysvětli aspoň tři chyby paketů!
- Jaké znáš techniky znova zasílání paketu?
- Uvažujme prostor sekvenčních čísel, který používá IP, maximální rychlosť linky 2 Mbps,  $T = A = 500$  ms. Jaké je v tomto scénáři největší povolené MPL pro pakety velikosti 40 B?
- Odesilatel zaslal pakety se sekvenčními čísly 10 až 15, příjemce odpověděl NACK pro paket s číslem 12. Jak se zachová odesilatel, pokud bude používat algoritmus Go-Back-N a jak v případě selektivního znova zaslání?
- Jak se určuje hodnota retransmission timeoutu?
- Co je to řízení spojení, řízení toku a řízení zahlcení?
- Co je to slow start, congestion avoidance a fast recovery?
- Jaké služby transportní vrstvy nabízí UDP, TCP, MPTCP, SCTP, DCCP, QUIC?

## Architektura přepínačů

### Obecná architektura přepínače



## Základní části přepínače

- **Vstupní rozhraní** (fabric input interface)
  - Propojuje vstupní modul a přepínací logiku
  - Rozděluje příchozí pakety na buňky pevné délky pro přepínání
  - Odesílá data na přepínací logiku na pokyn plánovače
- **Vstupní buffer** (fabric input buffer)
  - Ukládá příchozí data, pokud je není možné poslat do přepínací logiky
- **Přepínací logika** (switch fabric)
  - Dynamicky propojuje síťové rozhraní a přenáší mezi nimi data.
  - Vytváří propojení pomocí přepínacích obvodů či sběrnic.
  - Implementována na základní desce
- **Plánovač** (scheduler)
  - Plánuje přepínání dat ze vstupu na výstup
- **Výstupní rozhraní** (fabric output interface)
- **Výstupní buffer** (fabric output buffer)

## Požadavky

- Maximální přenos dat přepínací logikou
- Paralelní přenosy mezi různými síťovými rozhraními
- Spravedlivé přidělování přenosového pásma
- Zachování pořadí paketů

## Metriky pro měření činnosti přepínače

- Propustnost: množství dat přenesených za jednotku času (bps či pps)
- Latence: doba přenosu paketu ze vstupního na výstupní rozhraní (s)
- Počet dostupných cest v přepínací logice, které propojují každou dvojici vstupních a výstupních portů (blokující vs. neblokující přepínání).

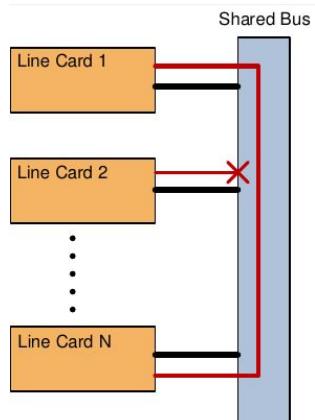
# Typy přepínačů

## Se sdílenou sběrnicí

- komunikuje v danou chvíli pouze jeden port
- s počtem portů roste šířka sběrnice
- vhodné do 1 Gbps
  - Potřebná propustnost sběrnice:  $R \times N$
  - Potřebná šířka sběrnice:  $w = \frac{R \times N}{r}$

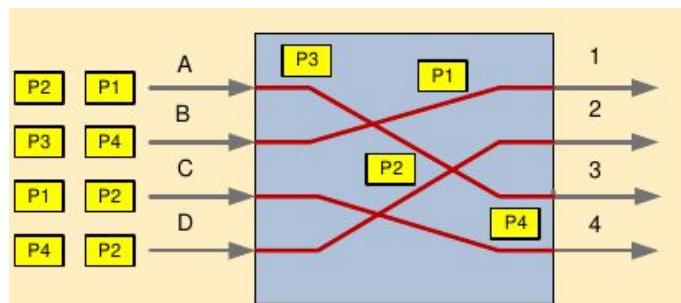
Přepínač má 16 portů o rychlosti 100 Mb/s, taktovací frekvence je sběrnice 40 MHz.

$$\bullet \text{ Propustnost } R \times N = 1,6 \text{ Gb/s} ; \text{ šířka sběrnice } w = \frac{R \times N}{r} = \frac{100 \cdot 10^6 \times 16}{40 \cdot 10^6} = 40 \text{ bitů}$$



## Přepínaná propojovací deska

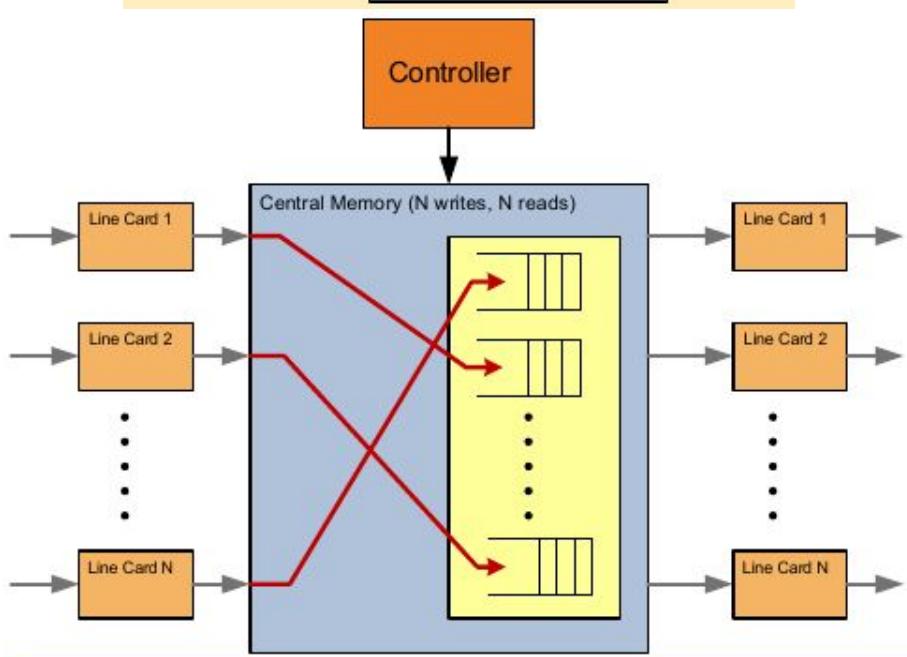
- paralelní přenos na více portech
- potřebuje plánovač
- Typy:
  - jednostupňové (shared mem)
  - vícestupňové (propoj. sítě)



## Jednostupňové

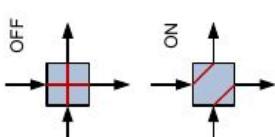
### Přepínač se sdílenou pamětí

- $BW = 2 \times N \times R [b/s]$
- $R$ : rychlosť sítového rozhraní
- $N$ : počet sítových karet
- centrální sdílená paměť
- př. str 310



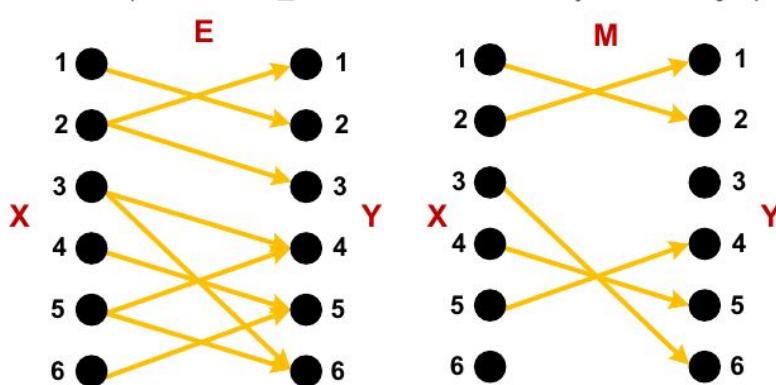
## Křížový přepínač (crossbar)

- $N^2$  propojení -> složitost při rozšíření
- neblokující - mohou vysílat paralelně
- potřebuje plánovač, který sítě přepíná



## Problém párování

- Hledáno takové párování  $M \subseteq E$ , kde žádné dvě hrany z  $M$  nemají společný vrchol



- Největší párování (maximum matching): párování, které má největší počet hran
- Maximální párování (maximal matching): párování, kdy nelze přidat další hranu

## Plánovací algoritmus přidělování lístků (příklad str 318)

### Princip algoritmu

- Výstupní port Q obsluhuje frontu požadavků na propojení.
- Požadavek na vstupu P dostane od Q číslo  $T_{QX}$  na obsloužení portu Q s pořadím X
- Výstupní porty jsou přiděleny žádostem s nižším  $T_{QX}$

### Fáze algoritmu

- ① Žádost o lístek (Request)
- ② Přidělení lístku (Grant)
- ③ Propojení vstupů a výstupů, přenos (Connect & Transfer)

### Zhodnocení algoritmu

- Umožňuje asynchronní zpracování
- Přenos rámců proměnné délky
- Blokování na začátku fronty (HOL, Head of Line Blocking)
- Nezávislé přidělování lístků: problém u multicastu

### Blokování na začátku fronty (HOL, Head of line)

#### Příčiny blokování

- Pakety na vstupních portech čekají ve frontě FIFO
- První buňka v pořadí blokuje další požadavky
- Blokované buňky čekají, i když je cílový port dostupný
- -> řešení: Rozdělení vstupní fronty na více virtuálních front podle cílových portů

### Virtuální výstupní fronty VOQ (Virtual Output Queues)

- Pro každý výstupní port vytvořena na daném vstupu jedna fronta
- Celkově potřebujeme  $N^2$  front

### Algoritmus PIM (Parallel Iterative Matching)

- Princip algoritmu
- Pracuje podobně jako algoritmus přidělování lístků, využívá virtuální fronty
- Hledá maximální párování pomocí náhodné volby požadavku ze vstupního portu
- Soutěžení o porty:
  - výstupní porty: více žádostí na jeden výstup
  - vstupní porty: více povolení pro data na jednom portu

### Fáze algoritmu

- ① Žádost o port (Request)
  - Vstupní port P posílá žádosti ze všech front na všechny žádané porty.
- ② Udělení portu (Grant)
  - Výstupní port Q přidělí právo přenosu.
  - V případě více žádostí o port Q, vybere *náhodně* jednu žádost.
- ③ Přijetí a přenos (Accept & Transfer)
  - Vstupní port P zpracuje udělená povolení k přenosu.
  - V případě udělení více povolení, *náhodně* vybere jedno. Dojde k přenosu dat.

## Algoritmus iSLIP

### Princip algoritmu

- Iterativní algoritmus pro hledání maximálního párování
- Při soupeření o port používá algoritmus *rotujících ukazatelů*:
  - $I_i$ : ukazatel na vstupním portu  $i$  (přijetí žádosti)
  - $O_j$ : ukazatel na výstupním portu  $j$  (udělení povolení k přenosu)
- Přidělení probíhá iterativně. Po každé iteraci se ukazatele inkrementují.
  - $I'_i = (I_i + 1) \bmod N$
  - $O'_j = (O_j + 1) \bmod N$
  - Ukazatele se inkrementují, až když je žádost o port potvrzena.

### Fáze algoritmu

- ① Žádost o port (*Request*)
  - Každý port pošle žádost na každý výstupní port
- ② Udělení portu (*Grant*)
  - Výstupní port Y vybere žádost, která má číslo portu větší nebo rovno ukazateli  $O_j$ ; v případě více žádostí se vybere žádost s nejmenší hodnotou
- ③ Přijetí volby (*Accept*)
  - Vstupní port X vybere povolení od portu, které je větší nebo rovno ukazateli  $I_j$ ; v případě více povolení se vybere povolení s nejmenší hodnotou

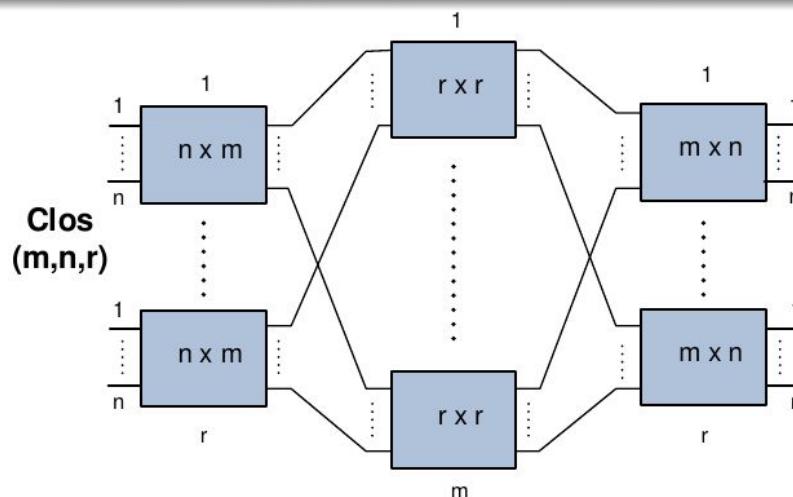
(str 330)

## Vícestupňové přepínání

- **Vlastnosti jednostupňového přepínání**
  - Využívá se převážně křížového přepínače (crossbar)
  - Problémy s kvadratickým nárůstem portů
  - Vnitřní blokování, blokování HOL
  - Součástí přepínání je hledání maximálního párování
  - Je možné zvětšit počet portů, aniž by se dramaticky rozšířilo přepínací pole?
    - Ano  $\Rightarrow$  s použitím vícestupňového přepínání.
- **Vlastnosti vícestupňového přepínání**
  - Sítě typu Clos a Beneš
  - vstup  $\rightarrow$  vnitřní přepínací obvody  $\rightarrow$  výstup
  - Vícestupňové obvody nemají vnitřního blokování.
  - Součástí přepínání je vytvoření bezkonfliktního propojení.

### Třístupňová přepínací síť Clos( $m, n, r$ )

- $r$  vstupních křížových přepínačů s  $n$  vstupy  $\Rightarrow$  přepínač má  $N = n \times r$  portů
- $m$  vnitřních křížových přepínačů s  $r$  vstupy  $\Rightarrow$  propojují každý V/V blok
- Mezi každým vstupním a výstupním portem existuje  $m$  různých cest.



### Closův teorém (neblokující síť)

- Pokud  $m \geq 2n - 1$ , pak lze přidat nové propojení vstupu a výstupu bez přeskládání.
- Tehdy je přepínač pro jakoukoliv konfiguraci vstupů a výstupů neblokující.
- Nevýhoda  $\Rightarrow$  síť vyžaduje velký počet vnitřních bloků.

### Vlastnosti sítě Clos(m,n,r)

- Přepínač s  $N$  vstupními porty potřebuje síť  $(m, n, \lceil N/n \rceil)$ .
- Pokud  $n = \sqrt{N/2}$ , pak minimální počet propojení je  $5.76 \times N \times \sqrt{N}$ .
- Počet propojení je stále lepší než křížový přepínač, tj. než  $N^2$ .

### Neblokující chování

- Pro neblokující chování musí platit Closův teorém, tj.  $m \geq 2n - 1$ .
- Například přepínač o 256 portech vyžaduje síť  $(31, 16, 16) \rightarrow$  drahé.

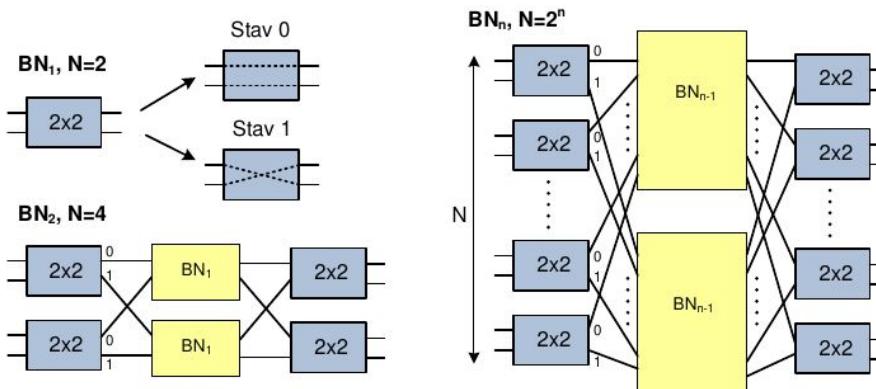
### Modifikované řešení: síť Clos(m,n,r), kde $m \geq n$

- Redukován počet prostředních přepínačů  $\rightarrow$  levnější.
- Síť není neblokující pro libovolnou konfiguraci vstupů a výstupů.
- Vlastnost modifikované sítě: síť je neblokující po přeskládání.
  - Výpočet přeskládání musí být rychlejší než přenos dat.
  - Používají se algoritmy barvení hran v bipartitním multigrafu.

### Přepínací síť Beneš

#### Rekurzivní přepínací síť Beneš $BN_n$

- Základem je modifikovaná síť Clos(2,2,1) s přeskládáním
- *Rekurzivní konstrukce sítě  $BN_n$* 
  - Počet vstupů (portů):  $N = 2^n$
  - Blok  $N/2$ -vstupních a  $N/2$ -výstupních přepínačů  $2 \times 2$
  - Prostřední část rekurzivních bloků  $BN_{(n-1)}$
  - Počet bloků (stupňů):  $2 \cdot \log_2(N) - 1$



### Propojování sítě

- Hledáme propojení – počet propojení je  $N!$
- Časová složitost algoritmů je  $\mathcal{O}(\sqrt{N})$  kroků.
- Používá se jednoduchý algoritmus využívající hierarchii sítě.

- Popište základní části přepínače a jejich funkci.
- Popište vlastnosti a chování přepínače se sdílenou pamětí.
- Jaké jsou výhody a nevýhody křížového přepínače?
- Co je to blokování HOL a jak ho lze eliminovat?
- Popište princip a použití algoritmu PIM.
- Popište princip a použití algoritmu iSLIP.
- Porovnejte algoritmy přidělování lístků, PIM a iSLIP z hlediska efektivity, implementace a spravedlivosti.
- Uvažujte čtyřportový křížový přepínač se vstupy A až D a výstupy 1 až 4. Na vstupu přepínače je fronta paketů s těmito cílovými porty: A → (1,3,3), B → (2,3,4), C → (3,2,1), D → (4,3,1). Zapište, jak budou vypadat přenosy v časových slotech 1-6 při použití planování za použití algoritmu (i) přidělování lístků, (ii) PIM, (iii) iSLIP.
- Popište princip a chování přepínací sítě Clos.
- Uvažujte přepínač o velikosti 128 vstupních portů. Kolik vnitřních propojení by vyžadoval při implementaci pomocí (i) křížového přepínače, (ii) neblokující sítě Clos, (iii) neblokující sítě Clos po přeskládání?
- Navrhněte architekturu 16-portového přepínače pomocí přepínací sítě Clos tak, aby bylo přepínání (i) neblokující, (ii) neblokující s přeskládáním.
- Zakreslete schéma přepínací sítě Clos (2,3,4).
- Popište architekturu a vlastnosti přepínací sítě Beneš.
- Zakreslete nastavení sítě  $BN_3$  pro následující konfiguraci vstupů a výstupů:  $\{(0, 2), (1, 6), (2, 5), (3, 0), (4, 7), (5, 1), (6, 4), (7, 3)\}$
- Zakreslete nastavení sítě  $BN_4$  pro následující konfiguraci vstupů a výstupů:  $\{(0, 0), (1, 4), (2, 8), (3, 12), (4, 1), (5, 5), (6, 9), (7, 7), (8, 2), (9, 6), (10, 10), (11, 14), (12, 3), (13, 7), (14, 11), (15, 15)\}$

## Teorie směrování

### Adresování

- pomocí IP, MAC, port
- Flat / Hierarchické / Pseudohierarchické (ipv6 - hierarch + if. id)

### Routování

#### IP směrovací tabulka (L3)

- nejlepší cesta zvolena podle metriky
  - různé vzorce výpočtu
  - nižší = lepší
  - možnost load-balancingu
- volení cesty podle admin. vzdálenosti
  - definovány "ceny" za průchod určitým typem rozhraní
  - čím méně a levnější rozhraní, tím rychlejší cena

Connected interface	0
Static route	1
Enhanced Interior Gateway Routing Protocol (EIGRP) summary route	5
External Border Gateway Protocol (BGP)	20

## CAM tabulka (content addressable memory) (L2)

- podle MAC se získá port, na který se mají data předat

# Routovací protokoly

## Typy

- **Distance- vector**
  - jedna metrika (není pravda .. viz EIGRP)
  - routování podle pověsti (sousedí si mění informace)
  - každý router rozhoduje kam data předá (na nejlevnějšího souseda)
  - nepotřebuje uniformní pravidla
  - Příklady: **RIP, EIGRP, Babel**
- **Link-state**
  - jedna metrika
  - každý zná topologii sítě a ceny
  - funguje jen pokud jsou ceny uniformní
  - každý router určuje další hop
  - Příklady: **OSPF, IS-IS**
  - **problémy se škálováním - problém mít v paměti celou topologii a výměny tabulek zatěžují síť**
- **Path-vector**
  - více metrik
    - preferovaný výstup / vstup AS
    - cesta v rámci AS
    - Původ paketu
  - flexibilní pravidla - různé cesty pro vstupní a výstupní traffic
  - síť jako skupina autonomních systémů
  - hledá se celá cesta, ne jen next-hop (ochrana před loop)
  - peering - předávání provozu mezi sousedními AS

## Distance-vector:

### RIP

- počet hopů jako metrika
- tabulky měněny pomocí UDP, max 25 cest v jednom
- bez detekce sousedů, prostě co 30s výměna
- počítání do nekonečna - pokud je někde výpadek, inkrementuje a předává tabulku dál, tím se ale vytváří smyčka, protože soused mu vrátí svou s inkrementovanou hodnotou a může počítat do nekonečna ~

### EIGRP

- nezávislé na L3
- classless
- kompozitní metrika (propustnost, zpoždění, spolehlivost, zátěž, spotřeba)
- každý má tabulku sousedů
- dvojitý KA řídící směrování
  - zaručuje trasu bez smyček v síti při směrování

### BABEL

- přepínatelné metriky, ...

## Link-state:

### OSPF

- "Hello" každých 5s - hledání sousedů
- Hierarchické
  - rychleji konverguje, méně zátěže sítě
  - dělení sítě na podoblasti pro zajištění hvězd
  - udržuje jen část topologie
  - oddělení interních a externích cest

- Distribuované a replikované
  - popisuje rovnou celou cestu
  - stejně pro všechny routery
  - lifetime tabulky 60 minut, refresh co 30
- Metrika "cena" = 100 Mbps / propustnost => (1 pro 100 MB a více)

### **IS-IS**

- předchůdce OSPF
- Dělá taky periodicky hello
- více tabulek různých metrik
- 4 metriky - defaultní, expense (cena za data po lince), zpoždění, počet chyb linky
- přímo encapsulovaný na L2 vrstvě
- Moc ho teda nechápu....

### **Path-vector:**

#### **EGP**

- AS (autonomous system) má zodpovědnost za informování o propojení do jiných AS
- Má mechanizmus, který umožňuje routerům mimo jádro získat informace o cestách
- classful
- Nespolehlivý přenost
  - sekvenční čísla
  - polling
- Vyžaduje stromovou topologii AS, aby nebyly smyčky

#### **BGP**

- TCP
- povoluje více jader sítě a propojení mezi AS
  - používá path-vector aby nevznikaly smyčky
- založen na pravidlech a složené metrice
- chyby se promítou do okolních AS až celého internetu
  - dnes používán skoro všude (AS)
- Externí BGP (mezi AS) / interní BGP (v AS)
- *Exchange entire BGP table first*
- *Later exchanges only incremental updates*
- *Application (BGP)-level keepalive messages*
- *Hold-down timer (at least 3 sec) locally config*

## Multicast routing protocols (NA PÍSEMCE NEBUDOU)

### **DVMRP**

- ???

### **PIM**

- spíš signalizační protokol
- potřebuje další protokol pro směrování, ale je na něm nezávislý (poskytuje mu informace)
- Režimy:
  - Dense - komunikace rozprostřena na celou topologii, pokud router nemá žádného zájemce o stream, odhlásí se z něj
  - Sparse - posíláno distribučním stromem, který je vytvořen z přijímajících klientů
  - Source-specific - IGMPv3 specifikuje multicast zdroj od kterého přijímá
  - BiDirect. - odesílatelé a příjemci komunikují

### **MSDP**

- multicast mezi AS
- v reálu moc nefunguje - extrémní nároky

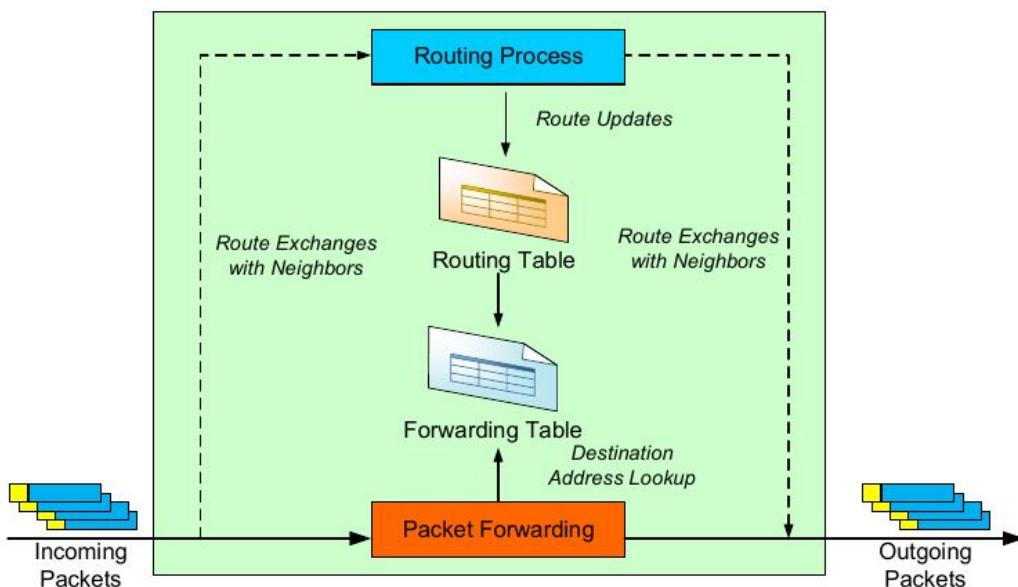
## Self-Check

- What is the advantage of using hierarchical addresses over flat ones?
- Does IP address identifies node or node interface?
- How distance-vector routing protocols models network topology?
- How link-state routing protocols models network topology?
- What is a difference between distance-vector and path-vector protocols?
- What is a difference between distance-vector and link-state routing protocols?
- What is a difference between unicast and multicast IP routing?
- Illustrate Bellman-Ford algorithm on example!
- Illustrate Dijkstra's algorithm on example!
- Compare RIP with EIGRP and Babel!
- Compare OSPF with IS-IS!

## Architektura směrovačů

Základní operace ve směrovači:

- Směrování (routing)
- Přepínání paketů (packet forwarding)



### Směrovací tabulka (routing table)

- IP prefix × cílové síť × sousední uzel (next hop)
- Optimalizovaná pro výpočet dynamických změn v topologii

### Přepinací tabulka (forwarding table)

- IP prefix × výstupní rozhraní × L2 adresy (výstupní rozhraní + sousední uzel)
- Optimalizovaná pro vyhledání cílové adresy

## Co patří mezi základní operace?

- Validace hlavičky L3 - Formát, verze, délka
- Kontrola hodnoty TTL a její dekrementace
- Přepočítání kontrolního součtu
- Zpracování rozšířených voleb IP protokolu
- Timestamp, record route, strict source route
- Vyhledání cesty (next-hop) na základě adresy
  - Lokální doručení, unicast, multicast
- Fragmentace a defragmentace IP datagramů
  - Pokud MTUin < MTUout
- Zpracování zpráv ICMP a IGMP

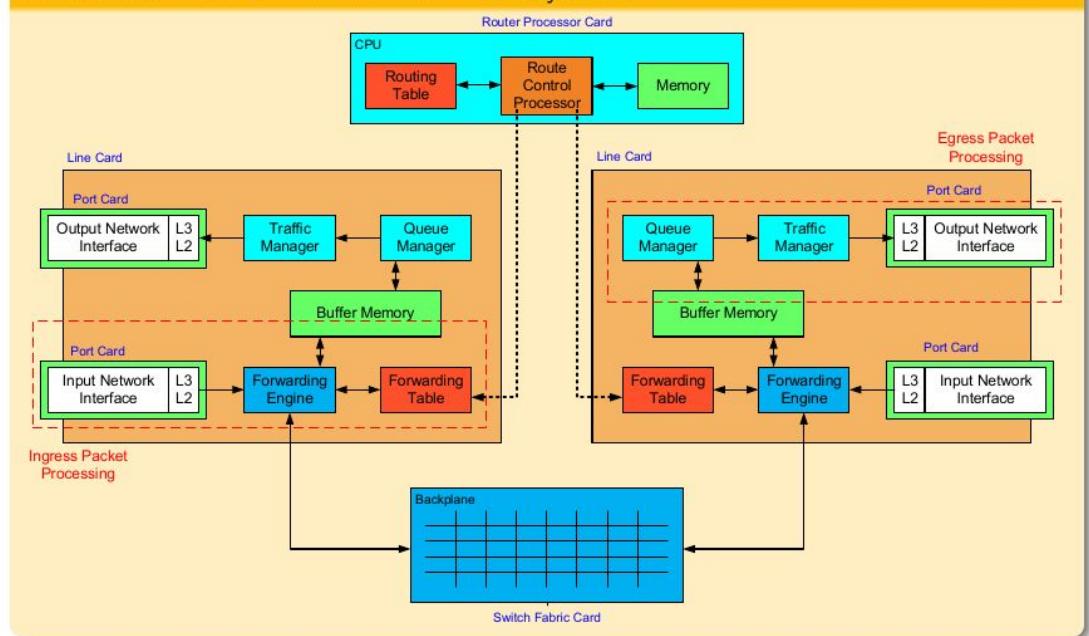
## Pokročilé:

- Dyn. směrování, filtrování, NAT, tunelování, klasifikace, DHCP, SNMP...

## Typy podle výkonnosti:

- Core router - až desítky Tb/s
- Edge router - typicky stovky Gb/s
- Enterprise - typicky desítky Gb/s
- SOHO - domácí použití kolem 1 Gb/s

Obecné schéma směrovače: funkční části a fyzické části



## Funkční části:

### Síťové rozhraní (Network Interface)

- Obsahuje více vstupních/výstupních portů
- Odstraní zapouzdření L2
- Předá hlavičku L3 přepínacímu modulu
- Uloží paket do paměti
- Zapouzdří odchozí pakety

### Řízení přepínání (Forwarding Engine, FE)

- Dostane hlavičku L3 ze síťového rozhraní
- Určí výstupní rozhraní podle informace ve FIB
- Provádí klasifikaci paketů pro podporu QoS na výstupu

### Správce front (Queue Manager)

- Ukládá pakety do vyrovnávací paměti, pokud je výstupní port obsazen
- Spravuje výstupní frontu → různé typy výstupních front
- Při zaplnění fronty vybírá a zahazuje pakety podle definované politiky

### Správce provozu (Traffic Manager)

- Prioritizuje a reguluje výstupní provoz podle požadavků QoS
- Omezuje či ořezává výstupní provoz (shaping, policing)

### Propojovací deska, sběrnice (Backplane)

- Propojuje síťové rozhraní
- Vytváří sdílené (shared) či přepínací (switched) propojení
- Rychlosť přepínání odpovídá přenosovému pásmu všech rozhraní

## Fyzické části:

### Řídící procesor pro směrování (Route Control Processor)

- Implementuje směrování na obecném CPU
- Zpracovává směrovací informace: aktualizace, udržuje sousedství
- Obsluhuje směrovací tabulku
- Přenáší data do přepínací tabulky
- Zpracovává pakety, které nelze směrovat pomocí FIB
- Vytváří chybové zprávy ICMP

### Modul fyzického rozhraní (Port Card)

- Implementuje síťové rozhraní: Ethernet, SONET, xDSL
- Provádí operace nad L2: zapouzdřování, vypouzdřování
- Udržuje statistiky o odchozím a příchozím provozu

### Síťový modul (Line Card)

- Obsahuje přepínání, správu front a správu provozu
- Obsahuje paměť pro uložení zpracovaných paketů
- Analyzuje hlavičku IP, vyhledává výstupní rozhraní
- Připojen k základní desce (backplane)

### Přepínací pole (Switching Fabrics)

- Přenáší pakety ze vstupního rozhraní na výstupní
- Páteřní směrovače obsahující více modulů přepínacího pole

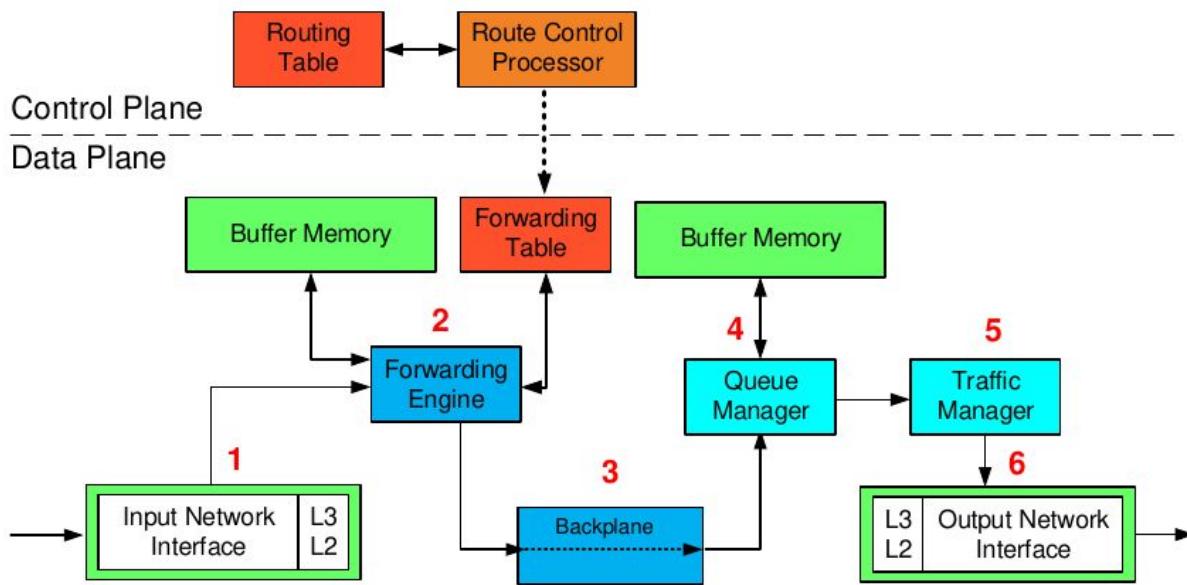
### Modul pro směrování (Route Processor Card)

- Implementace funkcí směrování a správy zařízení
- Běžící procesy pro směrování a správu
- Obsahuje obecný proces se specializovaným OS a velkou pamětí

## Zpracování paketu:

### Kontext

## Fáze zpracování

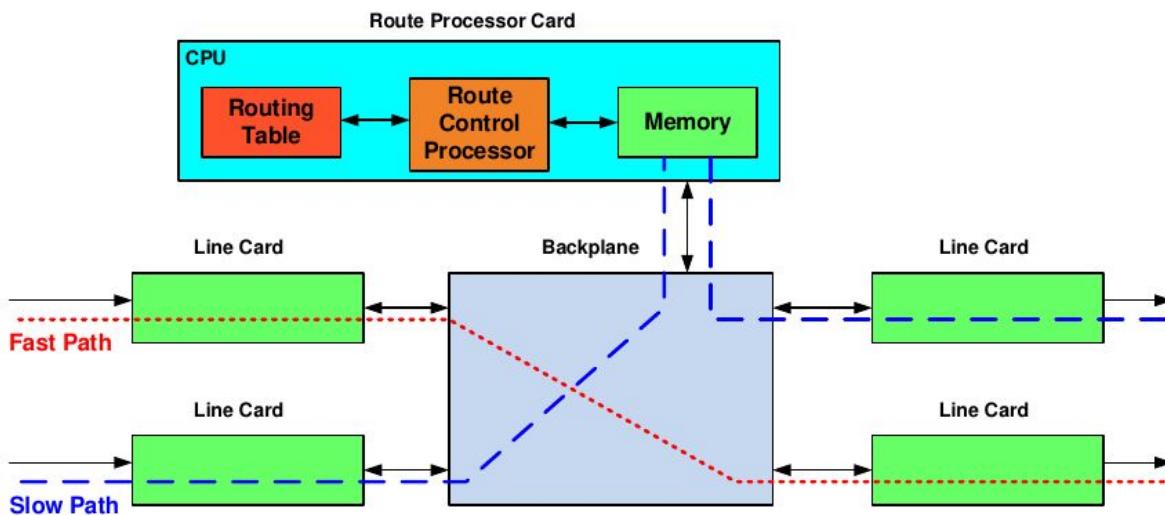


- směrovač si udržuje datovou strukturu obsahující vybrané položky z hlaviček L2 až L4

- 1. Paket přijde na síťové rozhraní
  - Síťové karta zpracuje L2 rámec, zkontroluje FCS
  - Vytvoří kontext paketu: vloží L2 zdrojovou a cílovou adresu
  - Zpracování hlavičky L3: typ protokolu, kontrolní součet, TTL
  - Doplnění kontextu o informace L3: IP adresy, typ protokolu, DSCP + porty
- 2. Zpracování v přepínacím modulu
  - Vyhledání cesty v přepínací tabulce: next hop + výstupní rozhraní
  - Doplnění dalších informací do kontextu paketu
  - Paket uloží do vyrovnávací paměti → adresa vložena do kontextu
- 3. Přeposlání kontextu propojovací deskou
- 4. Zpracování správcem front
  - Paket i kontext přeneseny na výstupní rozhraní
- 5. Předání kontextu správci provozu
  - Kontrola omezení rychlosti (shaping) dle kontextu
  - Překročení rychlost: zahodení či zpomalení
- 6. Výstupní síťové rozhraní
  - L3: aktualizace TTL, přeypočítání kontrolního součtu
  - L2: přidání hlavičky, výpočet CRC
  - Odeslání paketu (zapsání na výstupní médium)

### Rychlá cesta a pomalá cesta (Fast Path and Slow Path)

- Dva způsoby zpracování paketů ve směrovači
- Rychlá cesta (data plane) vs. pomalá cesta (control plane)



### Rychlá cesta (data plane)

- zpracování základních funkcí - hlavička, přeposlání paketu, klasifikace, předání do fronty, plánování
- řešeno v HW, maximální rychlosť

### Pomalá cesta (control plane)

- Zpracování softwarem - obvykle složitější operace - ARP, fragmentace, ICMP, SNMP

## Packet forwarding (přepínání paketů)

Přepínání paketů z jednoho rozhraní na druhé na základě směrovacích informací je jedna z nejdůležitějších funkcí směrovače.

Proces přepínání paketů zahrnuje:

- ① zjištění, zda cíl cesty paketu je dosažitelný,
- ② vyhledání nejbližšího uzlu na cestě (next-hop) a určení výstupního rozhraní,
- ③ vyhledání informací pro vytvoření L2 hlavičky paketu na výstupu.

⇒ Každý z těchto kroků je kritický pro úspěšné odeslání paketu.

Způsoby přepínání paketu

- 1) Softwarové přepínání (Process Switching)
- 2) Rychlé přepínání (Fast Switching)
- 3) Expresní přepínání CEF (Cisco Express Forwarding)

### Softwarové přepínání (Process Switching)

- Pro každý paket se hledá cesta ve směrovací tabulce a určuje se MAC adresa cíle.
  - Směrovač využívá standardní mechanismus přepínání procesů v OS.
1. I/O procesor detekuje paket na vstupním médiu. Přenese ho do vstupního bufferu.
  2. I/O procesor vygeneruje přerušení. Během přerušení určí centrální procesor typ paketu a zkopiuje ho do centrální paměti.
  3. Centrální plánovač zjistí, že ve vstupní frontě je paket. Naplánuje jeho další zpracování procesem ip input.
  4. Proces pro zpracování vyhledá ve směrovací tabulce další uzel (next hop) a výstupní rozhraní. V paměti ARP cache vyhledá MAC adresu dalšího uzlu.
  5. Přepíše L2 hlavičku paketu a umístí paket do výstupní fronty na výstupním portu.
  6. Paket vložen do fronty na výstupním portu.
  7. I/O procesor detekuje paket ve vysílací frontě. Zapíše ho na síťové médium.

Vyvažování zátěže (load balancing) při softwarovém přepínání

- Vyvažování v případě více cest ve směrovací tabulce probíhá po paketech.
- Pakety jsou automaticky rozděleny podle metriky (ceny) každé cesty.

### Rychlé přepínání (fast switching)

- Využívá vyrovnavací paměť route cache s předpočítanou L2 hlavičkou.
- První paket toku se přepíná softwarově. Další pakety toku jdou rychlou cestou.

Vyvažování zátěže (load balancing) při přepínání pomocí paměti Fast cache

- Vyvažování zátěže není po paketech, protože směrování je odděleno od přepínání.

## Vlastnosti rychlé paměti cache (Fast Cache)

- Paměť původně implementována pomocí hešovací tabulky → problém kolizí.
- Novější implementace využívají prefixového binárního stromu 2-way radix tree
  - Kompaktní varianta stromu prefixů *trie*
  - Počet potomků vnitřních uzlů  $\leq$  základ (radix)  $r$ .
  - Pokud uzel má pouze jeden list, připojí se list k předchozímu uzlu (redukce prostoru).
  - Hrany mohou obsahovat posloupnost bitů.

- 
- Inserting 2, 7 and 10
- Záznamy vkládány do paměti cache při přepínání paketu.
  - Rekurzivní odkazy se vyhodnotí před vložením záznamu do tabulky.
  - Neexistuje synchronizace mezi směrovací tabulkou, ARP cache a Fast cache.
  - Záznam se zneplatní při změně ARP cache či směrovací tabulky.
  - Při zaplnění paměti nad určitou mez se začnou záznamy náhodně zahazovat.

## Expresní přepínání CEF (Cisco Express Forwarding)

### Omezení rychlého přepínání (Fast Switching)

- Fast cache neumožňuje vložit masku → problém u překrývající se záznamů.
- Změny v ARP tabulce způsobují zneplatnění záznamů ve Fast cache.
- První paket se vždy zpracovává softwarově.
- Nedostatečné vyvažování zátěže.

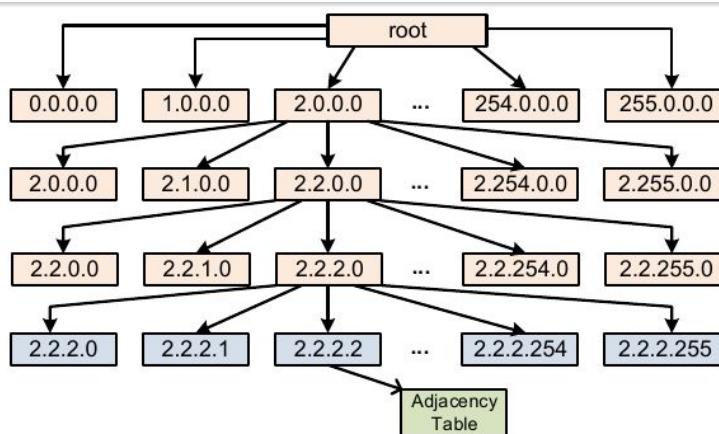
Nevadí u běžných podnikových směrovačů ⇒ kritické pro páteřní směrovače.

### Požadavky na páteřní směrovače

- Extrémně velké směrovací tabulky (sta tisíce záznamů)  
→ *Potřebujeme efektivní datovou strukturu pro uložení směrovací tabulky.*
- Dochází k častým změnám ve směrovacích tabulkách, což vede k zneplatnění záznamů ve Fast cache a výpadkům ve vyhledávání.  
→ *Potřebujeme oddělit směrování a přepínání.*
- Velká režie softwarového přepínání  
→ *Zkusme předpočítat tabulku pro přepínání předtím, než přijdou pakety.*

### Tabulka CEF

- Vytváří se dynamicky na základě směrovací tabulky.
- Optimalizována pro vyhledávání pomocí 256-ární struktury *trie* (256-way mtrie).
- Každý uzel může mít až 256 potomků reprezentující další byte adresy IPv4.
- List obsahuje ukazatel do tabulky sousedů (Adjacency Table).



### Tabulka sousedů (Adjacency Table)

- Obsahuje data pro vytváření L2 hlaviček pro přímo připojené sousedy:
  - MAC adresu cíle (next-hop), MAC adresu zdroje, číslo IP protokolu a další
- Vytváří se na základě ARP tabulky, mapovací tabulky Frame Relay, apod.
- Typy záznamů v tabulce sousedů:
  - Předpočítané hlavičky přímo připojených sousedů.
  - Nekompletní L2 hlavičky → vyžadují dotaz ARP.
  - Pakety určené pro softwarové zpracování.

### Výhody mechanismu CEF

- Tabulka CEF předpočítána na základě směrovací tabulky a tabulky sousedů ještě před příchodem paketu → nedochází k softwarovému přepínání.
- Oddělení směrovacích informací od L2 dat → nedochází ke stárnutí záznamů při expiraci záznamu v tabulce ARP.
- Změny ve směrovací tabulce či tabulce ARP se okamžitě propagují do tabulky CEF.
- ARP tabulka se synchronizuje se záznamy v tabulce sousedů.

### Vyvažování zátěže

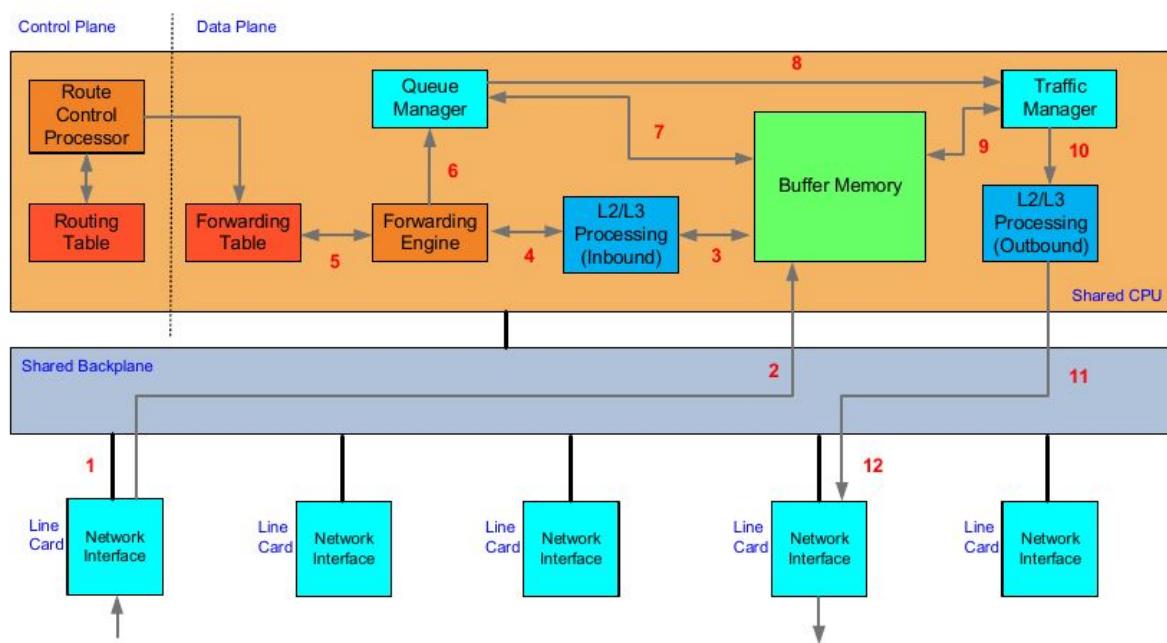
- Vyvažování lze provádět podle paketu nebo podle dvojice zdroj/cíl.
- Vyvažování podle dvojice zdrojová/cílová adresa:
  - Příslušný záznam v tabulce CEF ukazuje na tabulkou Load Share.
  - Implementována jako hešovací tabulka podle cílové a zdrojové adresy.
  - Obsahuje ukazatele na paralelní cesty v tabulce sousedů.

# Architektury směrovačů

## Architektury směrovačů podle způsobu přepínání paketů [2]

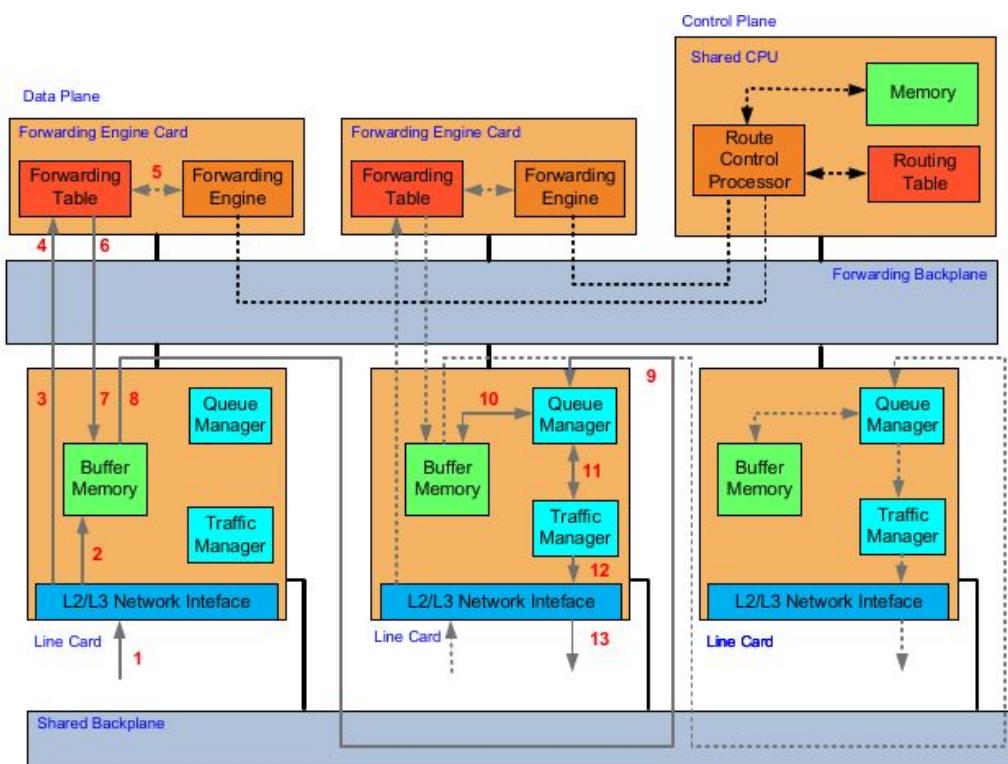
- ① Architektura se sdíleným procesorem (Shared CPU)
  - Varianta s vyrovnávací pamětí na kartě
- ② Architektura s nezávislými moduly FE (Shared Forwarding Engine)
- ③ Distribuovaná architektura (Shared Nothing)
- ④ Modulární architektura (Clustered Architecture)

### Se sdíleným CPU



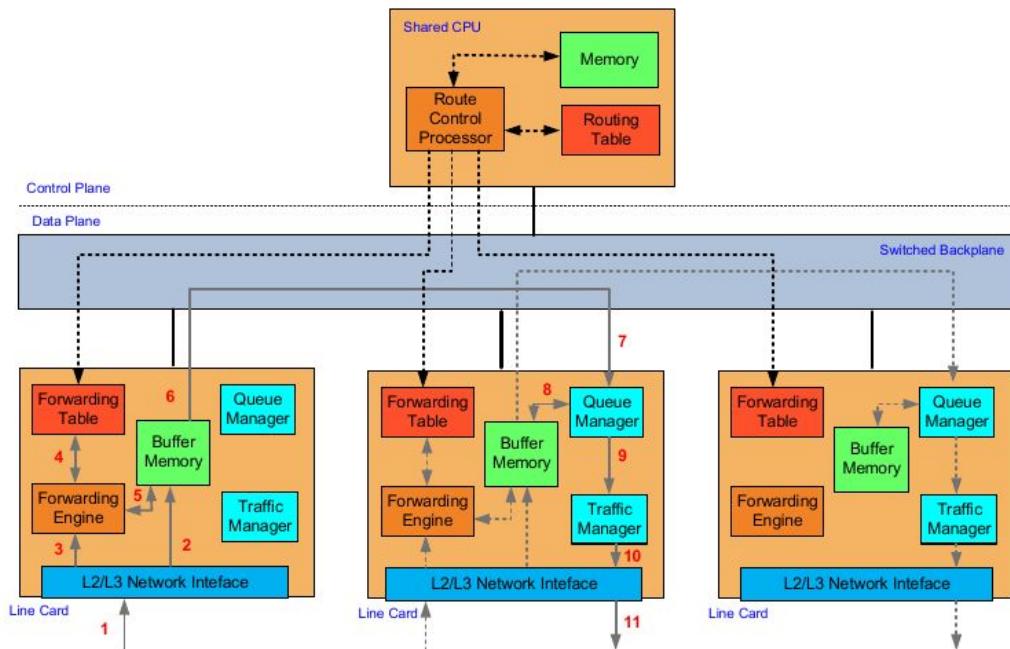
- Využívá softwarové přepínání → každý paket zpracován na CPU.
- Cykly CPU rozděleny mezi přenos, směrování a další operace.
- Sdílená sběrnice i procesor ⇒ levné, ale pomalé.
- Varianta s pamětí cache na kartě → synchronizace přepínacích tabulek.
- Sítová karta obsahuje FE pro zpracování hlaviček, paměť a přepínací tabulku.
- Rychlé přepínání (Fast Switching): první paket vs. další pakety.

# Architektura s nezávislými moduly FE



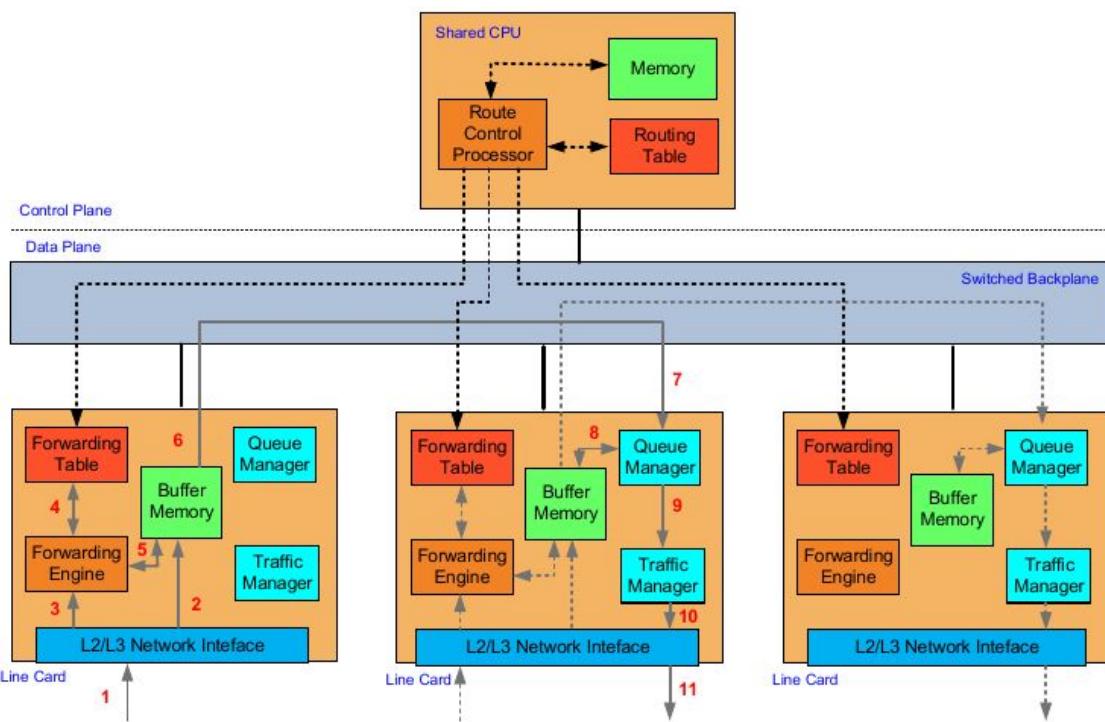
- Přepínací moduly FE implementovány na speciálních kartách.
- Paralelní zpracování paketů, dvě sběrnice (sdílená a přepínaná).

## Distribuovaná architektura



- Veškeré zpracování paketu přeneseno do sítového modulu.
- Oddělení procesu směrování a přepínání → využití technologie CEF.

## Modulární architektura (cluster architecture)



- Veškeré zpracování paketu přeneseno do síťového modulu.
- Oddělení procesu směrování a přepínání → využití technologie CEF.
- Nezávislé moduly připojené k centrálnímu přepínači.
- Více přepínacích modulů, více směrovacích procesorů.
- Distribuované zpracování → dCEF (distributed CEF).

## Otázky k opakování



- Vyjmenujte a stručně popište základní a pokročilé funkce směrovače?
- Popište základní stavební prvky (funkční moduly) směrovače?
- Co je to kontext paketu a k čemu se používá?
- Co jo rychlá a pomalá cesta při zpracování paketů ve směrovači? Popište průchod paketu pomalou a rychlou cestou a příklad zpracování.
- Popište softwarové přepínání na směrovači.
- Čím se liší rychlé přepínání (fast switching) a expresní přepínání (CEF) na směrovači?
- Popište průběh expresní přepínání paketů CEF ve směrovači. Jaké tabulky se používají, co obsahují a jak se aktualizují?
- Popište průchod paketu směrovačem u architektury se sdíleným procesorem, s nezávislými jednotkami FE a u distribuované architektury.

## Zpracování paketů

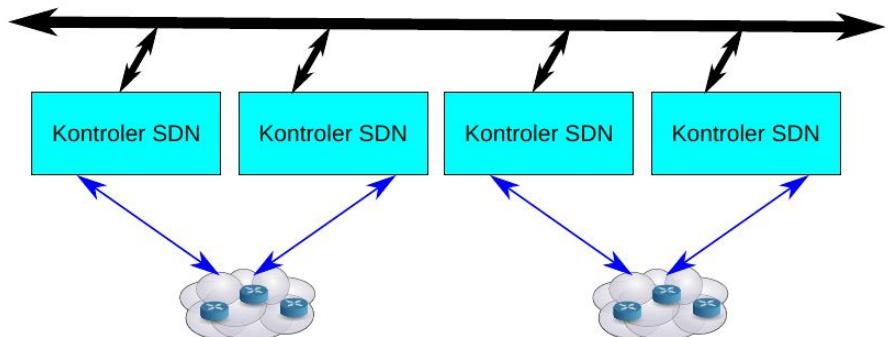
### OS

- vrstvy stejně jako v teoretickém modelu
- obvykle se zajímá jen o L2 - L4, zbytek řeší aplikace, nebo HW (L1)
- poskytuje API pro práci se síťovým zařízením a pakety

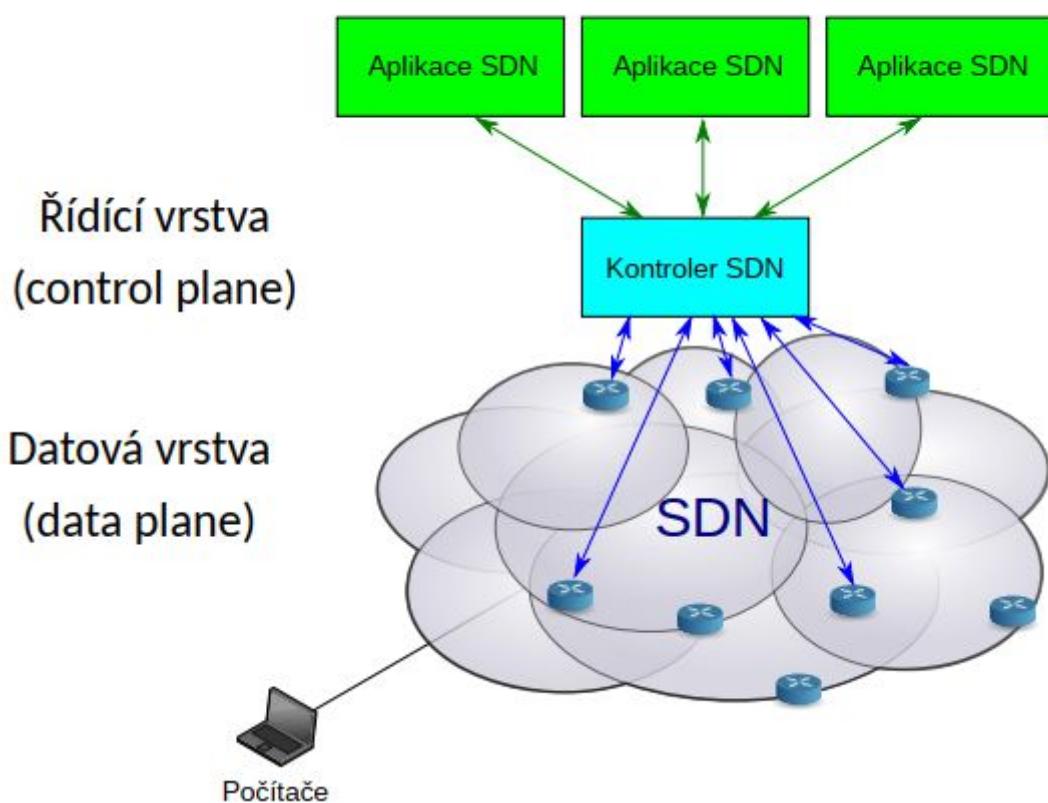
# SDN (software defined networking)

- upravují směrování v síti tak, aby byla optimálně využita celá síť
  - řízeno kontrolérem
- Využito k chytřejšímu řízení sítě pomocí softwarových pravidel (dle toků konfigurujeme zařízení)

## Řízení SDN



## Architektura SDN



## Kontrolér

- správa koncových uzlů
- správa topologie
- správa toků
- statistiky
- směrování
  - výpočty cest

## Vrchní rozhraní (Northbound)

- Každý kontrolér implementuje po svém
- špatná přenositelnost

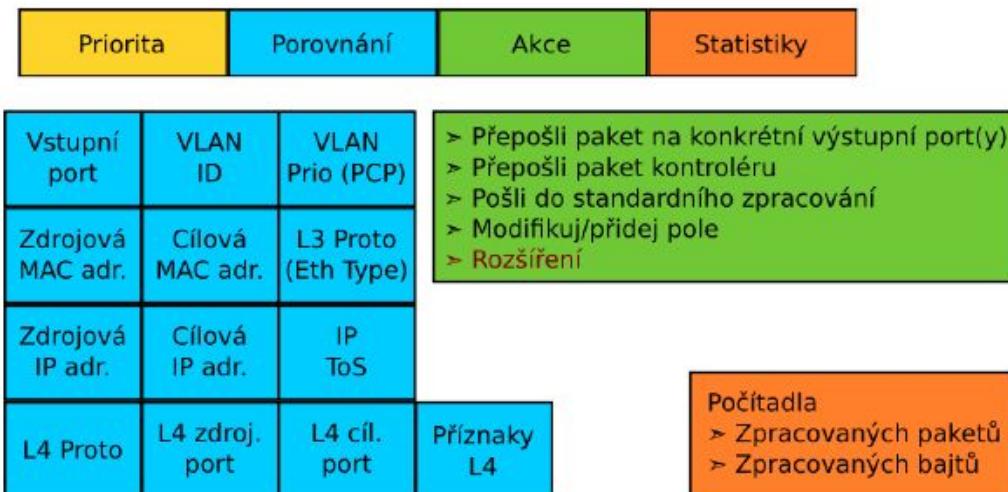
- REST a jiné protokoly
- stará se o řízení kontroleru - obvykle plugin

## Spodní rozhraní (Southbound)

- realizuje řízení zařízení (virtuální switch, firewall, balancer)
- OpenFlow a podobné

## OpenFlow

- binární protokol
- porovnává různé hodnoty v paketu a podle nich provádí akce a počítá statistiky



## Přepínání

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

\* všechno z této mac adresy na port 6, může to být třeba nějaká IDS / IPS, nebo sonda, honeypot. - zařízení se chová jako switch.

## Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

## Reaktivní

- první paket nového toku do kontroleru
  - kontroler vytvoří pravidla a uloží do tabulky prvku
- režie s komunikací
- typicky mikrotoky
  - kontrolér získává detailní informace o dění v síti

## Proaktivní

- pravidla předinstalovaná kontrolerem
- musí znát dopředu očekávané toky
  - typicky agregace

- 1) Nalezení pravidla s nejvyšší prioritou
- 2) Rozšiř množinu akcí/metadat, modifikuj paket
- 3) Volitelně aplikuj akce
- 4) Volitelně skoč do další tabulky

# Peer-to-Peer

## Definice sítě P2P

- *Dynamický soubor nezávislých uzlů (peers), které jsou propojeny a jejichž zdroje (objekty) jsou k dispozici ostatním uzlům v této síti.* [3, 4]

## Čím se liší P2P sítě od klasických sítí typu klient-server

- Jiná koncepce architektury ⇒ odlišná role uzlů
- Jiný způsob adresování ⇒ adresování obsahem
- Jiný způsob směrování ⇒ lokální rozhodování, specifická struktura sítí
- Další vlastnosti ⇒ decentralizovanost, samo-organizovatelnost

## Co mají P2P sítě podobné s klasickými aplikacemi službami?

- Vyžadují funkční IP infrastrukturu.
- Musí řešit adresování, směrování, zabezpečení a další.

## Příklady sítí P2P

- Komunikace elektronických zařízení: Universal Plug-and-Play (UPnP), Bluetooth
- Sdílení objektů: Napster, Gnutella, KaZaA, eDonkey, BitTorrent
- Komunikace mezi uživateli: Skype, IM
- Sdílení výpočetního prostředí: seti@home, PlanetLab

## Definice sítě P2P

- *Dynamický soubor nezávislých uzlů (peers), které jsou propojeny a jejichž zdroje (objekty) jsou k dispozici ostatním uzlům v této síti.* [3, 4]

- Zdroje: výpočetní výkon, přenosové pásmo, disková kapacita, zařízení (tiskárny)
- Sdílené zdroje jsou přímo přístupné všem uzlům, ty je nabízejí a zároveň využívají.
- Síť obsahuje prostředky pro připojení uzlu k síti, hledání a využití zdrojů, apod.

## Typy sítí P2P

- *Pravé sítě (Pure):* odebrání libovolného uzlu ze sítě nemá vliv na ztrátu schopnosti sítě poskytovat služby.
- *Hybridní sítě:* pro svou činnost využívají centrální uzel pro poskytování části nabízených síťových služeb.
  - Centrální bod slouží k autentizaci, indexování, inicializaci uzlu, apod.

## Samo-organizovatelnost

- Decentralizovaná topologie, kde uzly spolupracují na vytvoření a udržování.
- Každý uzel zodpovědný za svůj lokální stav a část informací (zdrojů).
- Uzly mají částečný pohled na topologii sítě: směrování na nejbližší sousedy.
- Podobně se chovají sítě MANET (Mobile Ad hoc Networks).

## Autonomní chování (samořiditelnost)

- Uzly se chovají dle svého nejlepšího rozhodování (sdílení zdrojů vs. free-riders).
- Rozhodování je lokální a nepredikovatelné  $\Rightarrow$  má vliv na topologii sítě, směrování, rozmístění objektů.
- Uzly se mohou chovat zlomyslně.
- Problém s ověřováním identity uzlů a důvěryhodnosti (decentralizované řízení).
- Možnost kolektivní zneužití zdrojů za špatným cílem.

## Spolehlivost

- Spolehlivost sítě roste s redundancí uzlů a informací.
- Kopie objektů jsou umístěny ve více uzlech.

## Životnost uzlu (churn rate)

- Doba života uzlu je krátká a neodhadnutelná  $\rightarrow$  problém s garancí služby.
- Závisí na subjektivním lokální rozhodnutí.
- Má vliv na směrování a vyhledávání: rychlosť odlivu zákazníků (churn rate) [5].

## Používá se jmenný prostor

- jména (id) by měla být přidělena uzlům tak, aby k sobě měli blízko (ideálně fyzicky i datově), komunikace se sousedy je pak rychlejší a síť efektivnější. (Používá se metrika.)

## Směrovací tabulka

Každý uzel obsahuje lokální směrovací tabulku  $R_p(V_p, E_p) \subseteq G$ , která je součástí globální konfigurace sítě. Tabulka obsahuje množinu sousedních uzlů a hran k nim.

- Pro množinu sousedních uzlů platí:  
 $V_p \subseteq V, \forall (p, q) \in E \implies q \in V_p \wedge \exists q \in V_p : (p, q) \in E$ .
- Pro množinu hran platí:  $E_p \subseteq E, \forall (p, q) \in E \implies (p, q) \in E_p$ .

## Nestrukturované P2P

### 1) Záplava (flooding)

- Uzel pošle dotaz všem svým sousedům
  - Pokud soused obsahuje objekt, pošle zpět odpověď
  - Pokud nemá objekt, pošle zprávu svým sousedům (tranzitivita)
- Záplavu je možné omezit pomocí TTL ve zprávě
- Využívá např. Gnutella

### 2) Rozšiřující se kruh (expanding ring)

- Podobné jako záplava. Pošlu dotaz na objekt s malým TTL.
  - Pokud objekt najdu, hledání končí.
  - Pokud objekt nenajdu, zvýším TTL a pošlu dotaz znovu.
- Redukuje počet zpráv v síti

### 3) Náhodný průchod (random walk)

- Zpráva poslána náhodně vybraným sousedům
  - Možné poslat více sousedům současně
  - Potřeba kontrolovat vybraného souseda: neposílat dotaz zpět

### 4) Hledání lokálního minima (Local Minimum Search, LMS) [10]

Definujme tento algoritmický problém:

- Máme množinu uzlů identifikovaných hodnotou  $x$ .
- Máme množinu objektů s identifikátorem  $w$ 
  - Např. hash veřejného klíče, hash jména objektu
- Úkolem je umístit objekty do sítě uzlů tak, aby bylo možné najít rychle a spolehlivě, tj. jméno uzlu  $x$  by mělo být co nejblíže jménu ukládaného objektu  $w$ .

## Porovnání směrovacích algoritmů u nestrukturovaných sítí P2P

- Metoda záplavy a metoda rozšiřujícího se kruhu
  - Jednoduchá implementace
  - Minimální paměťové i výpočetní nároky
  - Neefektivní, špatně rozšiřitelná
- Metoda náhodného průchodu
  - Hledání bez znalosti umístění objektu
  - Nalezení objektu může dlouho trvat
- Metoda lokálního minima
  - Přidává znalost: metriku vzdálenosti k objektu
  - Směrování podle jména objektu
  - Vyžaduje režii při hledání lokálních minim

## Strukturované P2P

### Strukturované sítě

- Kombinují geometrické struktury a směrování.
- Využívají distribuované směrovací algoritmy:
  - Metriky: shoda prefixu, euklidovská či lineární vzdálenost, XOR, atd.
  - Velikost směrovací tabulku ovlivněna stupněm uzlů.

### Vlastnosti sítě BitTorrent

- P2P síť pro sdílení souborů.
- Síť využívá 160bitové identifikátory pro identifikaci uzlů (peer ID) i pro identifikaci sdílených souborů (`info_hash`).
- Více implementací: [Mainline DHT \(MLDHT\)](#), KAD, [VUZE](#) a další.
- Pro distribuci obsahu se využívá komunikace pomocí trackeru nebo distribuovaná hešovací tabulka DHT.

### Komunikace pomocí trackerů.

- Swarm [roj] = množina uzlů zapojených do sdílení souboru.
- Tracker [stopař] = uzel, který udržuje seznam uzlů zapojených do distribuce daného souboru.
- Torrent [příval] = soubor s informacemi o sdílených souborech (či adresářích) a s odkazem na tracker.
- Protokol BitTorrent = aplikační protokol pro distribuci souborů (nad TCP).

### Zhodnocení strukturovaných sítí P2P

- Klíčovým parametrem je geometrie (struktura) sítě.
- Potřeba implementovat operace pro připojení, odpojení, vyhledání.
- Směrovací algoritmus musí konvergovat k cíli.
- Nutné sledovat stav sousedů a pravidelně aktualizovat směrovací tabulku, případně tabulky sousedů.

### Příklady strukturovaných sítí

- Kadmelia
- BitTorrent
- Skype

### Směrování v Kadmelia

- Směrovací tabulka se aktualizuje dynamicky při příchodu zprávu od jiného uzlu.
- Pokud daný řádek obsahuje méně než  $k$  položek, přidá se nový uzel.
- Pokud je řádek plný, otestuje se dostupnost nejpozději přidaného uzlu. Pokud je nedostupný, nahradí se novým. Pokud je dostupný, nový uzel se do tabulky nepřidá.  $\Rightarrow$  preferují se starší kontakty.

## Otázky:

- Vysvětlete experiment Stanleyho Milgrama a jeho vliv pro návrh sítí P2P?
- Definujte sítě P2P. Popište jejich vlastnosti, výhody a nevýhody při použití.
- Popište rozdíly architektur sítí P2P a klient-server.
- Popište referenční model sítí P2P.
- Čím se liší strukturované a nestrukturované sítě P2P.
- Jaké znáte směrovací algoritmy u nestrukturovaných sítí P2P?
- Popište metodu hledání lokálního minima.
- Popište směrování v síti Kadmelia.
- Popište chování sítě BitTorrent za pomoci trackerů a pomocí DHT.

## Identifikace síťového provozu a důvěra na Internetu

- Bezpečnost: detekce útoků, nedovolené chování uživatelů a aplikací
- Diagnostika: správné chování systémových i uživatelských aplikací
- Rozlišení služeb: nastavení kvality služeb, prioritizace kritických přenosů
- Vytížení sítě: sledování rozložení provozu, vytížení síťových prvků a služeb
- Účtování služeb: identifikace provozu VoIP, IPTV, apod.

### Způsoby identifikace:

1. Typická data z hlaviček paketů
2. Identifikace podle signatury v obsahu paketů (DPI)
3. Statistické chování toků

#### Jaké hlavičky protokolů lze použít?

- Hlavička Ethernetu (EtherType)
  - IPv4 (0x0800), ARP (0x0806), RARP (0x8035), IPv6 (0x86DD), PPP (0x880B), MPLS (0x8847), 802.1x (0x888E), 802.1q (0x88A8), ...
- Hlavička IP (protocol)
  - ICMP (1), IPv6 (41), IGMP (2), UDP (17), TCP (6), SCTP (132), GRE (47), ESP (50), AH (51), RSVP (46), EIGRP (88), OSPF (89), PIM (103), L2TP (115) ...
- Hlavička TCP/UDP (Source Port, Destination Port)
  - FTP (20,21), SSH (22), Telnet (23), SMTP (25), DNS (53), DHCP (67, 68), TFTP (69), HTTP (80), POP3 (110), NNTP (119), NTP (123), IMAP (143), SNMP (161,162), BGP (179), LDAP (389), HTTPS (443) ...
- Hlavička aplikacního protokolu
  - HTTP: "GET / HTTP/1.1"
  - BitTorrent: "0x13BitTorrent protocol"

### Typická data z hlaviček paketů

- nemusí vždy fungovat:
  - nemusí být svázána s číslem portu
  - tunelování, zapouzdření
  - maskování v jiném protokolu (ICMP, DNS)

### Identifikace podle signatury v obsahu paketů (DPI)

- hledáme typické řetězce

- na dané pozici
- v rámci celého paketu
- obvyklé posloupnosti
- Někdy to nejde - proprietární protokoly, nové protokoly, šifrované, fragmentace
- Použití:
  - vytvoření databáze signatur
  - hledání v databázi

### **Omezení při použití signatur**

- Definice signatury vyžaduje expertní znalost
  - Sémantická analýza daného protokolu
  - Empirické zkoumání obsahu paketu a hledání signatury
- Potřeba vytvářet a udržovat aktuální databázi signatur (obvykle stačí prvních 10 paketů)
- Časově i výkonnově náročné vyhledávání signatur v těle procházejících paketů

### **Automatické určování signatur: srovnání s jinými postupy**

- Paketová analýza: zkoumá obsah jednotlivých paketů nezávisle
- Protokolová analýza: manuální sémantická analýza protokolu

### **Statistické chování toků**

- Vytvoříme statistický model protokolu
  - Nevyžaduje data z hlaviček či obsahu paketů → stačí meta data o provozu.
  - Hledáme charakteristiky (atributy) popisující chování protokolu.
  - Všímáme si typického chování protokolu: odezva, velikost paketu, počet paketů v daném směru, apod.
- Klasifikujeme neznámý protokol oproti vytvořeným modelům protokolů  
⇒ Vybere se model s nejlepší vzdáleností od neznámého protokolu

### **Co potřebujeme k vytvoření statistického modelu protokolu?**

- Model aplikačních protokolů vytvořený z trénovací množiny dat
- Vlastnosti (atributy) toku pro statistické porovnání
- Metriku blízkosti pro porovnání modelu protokolu s klasifikovaným provozem
- Stanovení prahové hodnoty, kdy ještě může být protokol úspěšně identifikován
- Algoritmus porovnání (klasifikace)

### **Využití otisků protokolu (protocol fingerprints)**

- Otisk protokolu (protocol fingerprint) = soubor vlastností (chování) protokolu, které lze využít k jednoznačné identifikaci protokolu.
- Otisky protokolů vytvářejí modely protokolů pro statistickou klasifikaci.
- Narozdíl od signatur nezkoumáme obsah hlaviček či těla protokolu, ale pouze statistické vlastnosti jeho chování.
- Tyto vlastnosti lze využít k detekci protokolu.

⇒ Pro detekci stačí sledovat pakety během první fáze vytváření spojení.

### **Metoda jednoduchých statistických otisků [3]**

- Pro vytvoření databáze otisků využívá tři atributy:
  - Velikost IP paketu s (packet size)
  - Časový odstup paketů  $\Delta t$  (inter-arrival time)
  - Pořadí paketu v toku i (packet order)
- Klasifikace počítá vzdálenost paketu i od známého vzorku  $A_i$  (anomaly score).

## Popis metody

- Tok  $F$  (flow): jednosměrná sekvence  $n$ -paketů ( $P_1, P_2, \dots, P_n$ )
- Paket  $P_i$  popsán velikostí  $s$  (size) a časovým odstupem  $\Delta t$  od paketu  $P_{i-1}$ :  $P_i = \{s_i, \Delta t_i\}$

### (i) Učení = vytváříme otisku daného protokolu

- ① Počítáme hustotu rozdělení pravděpodobnosti  $PDF_i$  (Probability Density Function)
  - $PDF_i$  popisuje rozložení hodnot  $s$  a  $\Delta t$  pro každý  $i$ -tý paket zkoumaných toků.
  - Pro toky o délce  $L$  paketů tvoří vektor  $PDF$  složený z  $PDF_i$
- ② Aplikací Gaussova filtru odstraníme šum a vytvoříme masku protokolu  $M_i$  pro klasifikaci.

### Co lze použít jako atribut pro rozlišení aplikačního protokolu?

- Bytová frekvence prvního paketu TCP v každém směru včetně obsahu
- Počet změn směru komunikace: klient → server, server → klient
- Počet bytů v daném směru: RTP (balanced), HTTP (download), SMTP (upload)
- Entropie prvního paketu v každém směru
- Opakovaný výskyt dvojic stejných bytů v prvním paketu (TT, SS)
- Bitová frekvence prvních 128 bitů u UDP
- Velikost obsahu prvního paketu toku: 120-1000 B u HTTP, 10-100 B u POP3

Obvykle stačí k určení úvodních 10 paketů (pro udp 20), pro trénování databáze cca 30 toků.

### Srovnání metod:

- **Identifikace podle hodnot z hlaviček paketů**
  - Jednoduché na implementaci, rychlé
  - Nevyžaduje trénovací množinu
  - Identifikace na základě přesné shody
  - Omezená míra použitelnosti
  - Pro identifikaci se využívají čísla portů či čísla protokolů
- **Identifikace podle signatur**
  - Nespoléhá na hodnoty z hlaviček paketů
  - Vyžaduje vytvoření databáze signatur a její pravidelnou aktualizaci
  - Výpočetně náročné
  - Např. L7 filter, snort, Cisco IOS IPS a další
- **Identifikace podle statistického chování protokolu**
  - Vhodné pro tunelovaná či šifrovaná data
  - Využívá statistické vlastnosti toků
  - Zkoumá pouze několik prvních paketů toků
  - Vyžaduje vytvoření modelu protokolu na základě trénovací množiny.
  - Méně náročné než použití signatur
  - Může chybně detektovat provoz (false positives)

## Důvěra v internetu

### Problém důvěry

Problém důvěry v Internetu (provoz, uživatelé, zdroje) ⇒ snaha vytvořit globální mechanismu důvěry.

- **Emaily:** spamy, podvržené emaily, phishing
- **Webové služby:** podvržené stránky, stránky s malwarem, nevhodný obsah
- **Sdílení softwaru:** zavírované soubory, malware

- **Sítové spojení:** ochrana sítí proti zneužití (botnety, útoky DDoS)
- Důvěřující nemůže odhadnout důvěryhodnost poskytovatele důvěry přímo, ale hledá tzv. znaky důvěryhodnosti (tzv. signál).
- Co může být tím signálem?
  - Signál = aktivita nebo vlastnost, kterou může jednoduše splnit důvěryhodná entita, ale jejíž získání je pro nedůvěryhodnou entitu příliš drahé.

### Reputace

*Reputace (reputation) = znak důvěryhodnosti vyjádřený svědectvím dalších entit.*

- Efektivní rozlišující signál, který podporuje spolupráci postavenou na důvěře.
- Reputace není dokonale rozlišující signál.
- Reputace je efektivní, pouze jsou svědectví nezávislá a důvěryhodná.
- Reputace je založena na zkušnostech, historii či vztazích.

### Co je potřeba pro budování důvěry na Internetu? [10]

- Informace vhodné pro měření důvěry a reputace v dané aplikaci
- Metrika pro výpočet hodnoty reputace (reputation score) či míry riziku (risk rating)
  - Hodnocení závisí na aktuálních i historických informacích.
- Způsob získání a udržování těchto informací
- Reputační systém, který je odolný vůči manipulaci

### Centralizovaný reputační systém

- Hodnotitelé předávají hodnocení centralizovanému systému.
- Systém vyhodnocuje reputaci na základě vstupních hodnocení a dalších informací.
- Příklad: hodnocení reputace na základě hodnocení transakcí

### Jak počítat hodnotu reputace?

- ① Součet jednotlivých hodnocení
  - eBay's Feedback Forum:  $R = \sum_i positive\_score_i - \sum_j negative\_score_j$
  - Co je lepší: 100 pozitivních a 10 negativních ohlasů či 90 pozitivních ohlasů?
- ② Průměr jednotlivých hodnocení
  - Amazon: počítá hodnotu na stupnici 1-5, bere průměr hodnocení
- ③ Bayesovské systémy
  - Vstupem je binární hodnocení (pozitivní či negativní).
  - Reputační skóre se počítá pomocí pravděpodobnostního rozdělení Beta.
  - Hodnocení ve formě  $(\alpha, \beta)$ ,  $\alpha$  je počet pozitivních hodnocení a  $\beta$  negativních.
- ④ Modely důvěry (Belief Models)
  - Model počítá metriky důvěry na základě důvěry, nedůvěry a nejistoty.
  - Tyto názory se mapují na funkci hustoty pravděpodobnostního rozdělení Beta.
- ⑤ Modely toků
  - Počítají důvěru a reputaci na základě tranzitivity hodnocení účastníků.
  - Například algoritmus PageRank (Google), Appleseed či Advogato.

Reputace je služba založená na ohodnocení → není to černá listina. Reputace vyjadřuje, jak "riskantní" je komunikovat s daným uzlem. Reputace je časově podmíněná.

### Sítový reputační systém

- Sítový reputační systém poskytuje platformu pro monitorování aktivit v síti.
- Sbírá data ze sítových senzorů, firewallů, IPS zařízení, anti-virové analýzy apod.
- Provádí korelace dat a počítá hodnotu reputace či míru riziku
- Výpočet bere v úvahu historická data.

### Jaká data lze použít pro výpočet reputace?

- IP adresa a port útočníka
- IP adresa a port oběti
- Maximální velikost segmentu, volitelné hodnoty TCP, velikost IP datagramu
- Počet žádostí o příchozí či odchozí spojení
- Otisk správy, seznam spammerů, URI, záznamy DNS a další

## Otzázkы k opakování



- Popište klasifikaci toků podle hodnot z hlaviček rámců, datagramů a paketů. Jaké položky lze použít pro klasifikaci? Jaké jsou výhody a nevýhody tohoto postupu?
- Co jsou to signatury protokolů? Uveďte příklad. Popište postup automatizovaného vyhledání signatury.
- Co je to otisk protokolu a jak ho lze získat? Vyberte si nějakou metodu pro získání otisku protokolu a popište ji.
- Jaké atributy lze použít pro vytvoření statistického modelu protokolu? Uveďte alespoň pět atributů.
- Popište využití frekvenční analýzy při klasifikaci provozu podle statistického rozložení.
- Co je to důvěra na Internetu. Definujte tento pojem a uveďte, jak se zjišťuje a k čemu se využívá.
- Vysvětlete, co je to reputační systém a na základě čeho pracuje. Uveďte příklad využití v komunikaci na Internetu.
- Co je to reputační skóre a jak se vypočítá? Uveďte několik příkladů možného výpočtu, jejich výhody a omezení.
- Navrhněte reputační systém pro vybranou sítovou službu, způsob výpočtu reputačního skóre a zdroj dat pro tento výpočet.

# Anonymita

- IP adresa může být svázaná přímo s uživatelem
  - ISP si ukládá informace o komunikaci
  - Uloženy typicky po určitou dobu (Data Retention)
  - Law enforcement agency
- Browser fingerprinting
  - Cookies, Flash cookies, E-Tags, HTML5 Storage
  - Browser fingerprinting
    - javascript
- User fingerprinting
  - Aktivity uživatele – které aplikace používá, na které weby přistupuje

## Unlinkability

- Neschopnost spojit dvě události
- Např. pakety, přístupy na web, lidi, akce

### Tři části:

- Sender anonymity (Kdo to poslal?)
- Receiver anonymity (Kdo je příjemce?)
- Relationship anonymity (Jsou A a B v nějakém spojení?)

## Unobservability

- Nelze rozlišit monitorované události od jiných

## IPv4 adresa – přidělení uživateli

- DHCP, PPPoE
- ISP si uloží informaci kdo žádal (MAC, DHCP82, username) a jaká adresa byla přidělena

## Symetrické kryptografie

- Algoritmy se sdíleným klíčem, který se používá jak pro šifrování, tak pro dešifrování
- Stejný klíč znají obě strany, bezpečnost spočívá v ochraně klíče
- Výhody:
  - rychlé, jednoduché
- Nevýhody:
  - distribuce klíčů
- Alg: DES, AES

## Asymetrická kryptografie

- Používá se dvojice navzájem svázaných klíčů – veřejného a privátního
- Oproti symetrickým algoritmům je délka klíče mnohem větší k zajištění stejné míry zabezpečení
- Asymetrické algoritmy jsou náročné na výpočetní prostředky (**100x až 1000x pomalejší**)

### Privátní a veřejný klíč

- Privátní klíč zná a vlastní pouze majitel
- Veřejný klíč je k dispozici komukoli
- Oba klíče jsou rozdílné a je výpočetně „nemožné“ odvodit z jednoho klíče druhý
- Každý z klíčů může být použitý jak pro šifrování tak dešifrování
  - privátní šifruje, veřejný dešifruje
  - veřejný šifruje, privátní dešifruje

### Autentifikace

- pokud lze zprávu dešifrovat veřejným klíčem, tak je jisté, že ji odeslal vlastník privátního klíče

### Důvěrnost

- pokud zašifruji zprávu mým privátním a cizím veřejným, tak ji může dešifrovat jen vlastník cizího privátního s použitím mého veřejného

## Hash

- Jednocestná matematická hash funkce bere na svém vstupu binární data libovolné délky a produkuje výstup fixní délky zvaný hash
- Hash se používá k zajištění integrity
- Hash funkce by mělo být co nejvíce odolná kolizím

## VPN

- udělá se tunel na stroj s jinou IP adresou, tím se za něj schovám
- poskytovatel ví kdo se připojil a kam, ostatní jen půlku (znám zdroj, nebo cíl)

## Proxy

- pošlu příkaz, on se vykoná a vrátí se mi data
- proxy ví všechno a může krást cookie, data... cokoli

## Mix proxies a Onion routing

- pošlu data přes několik VPN, proxy a všechno šifrují, těžko potom něco dohledat.

## Traffic mixing

- Prevence timing útoků - data od uživatelů pozdrží náhodnou dobu, nelze pak určit jejich "vzdálenost"

## Cover Traffic

- schovám se ve smetí
- vytvářím nesmyslný provoz a snažím se schovat opravdový na pozadí

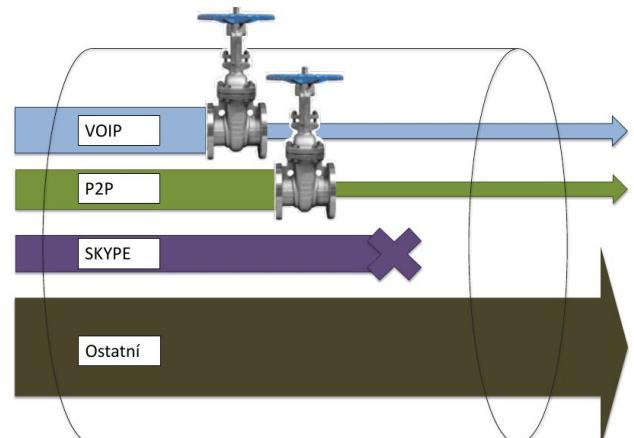
## TOR

- chová se jako SOCKS proxy
- používá mix servery (tor relay)
- relay servery uloženy v seznamu u důvěryhodných serverů
- pakety děleny na buňky 512B
- guard relays - vstupní brány do TOR vyměněny co 3 měsíce

## Síťová neutralita

### Ceny podle linky / přenesených dat

- autonomní systémy si určují ceny za přenos dat skrz jejich síť, ceny jsou buď dle rychlosti linky (20k / 1 Gbps), nebo podle množství (platí standardně 1 Gbps, ale když jedu rychleji, tak za příplatek)
- vytváří konkurenční prostředí díky různým cenám, rychlostem a stabilitě AS



## Neutralita

- provideri by se měli chovat ke všem datům stejně

## Formální metody

### Proverif

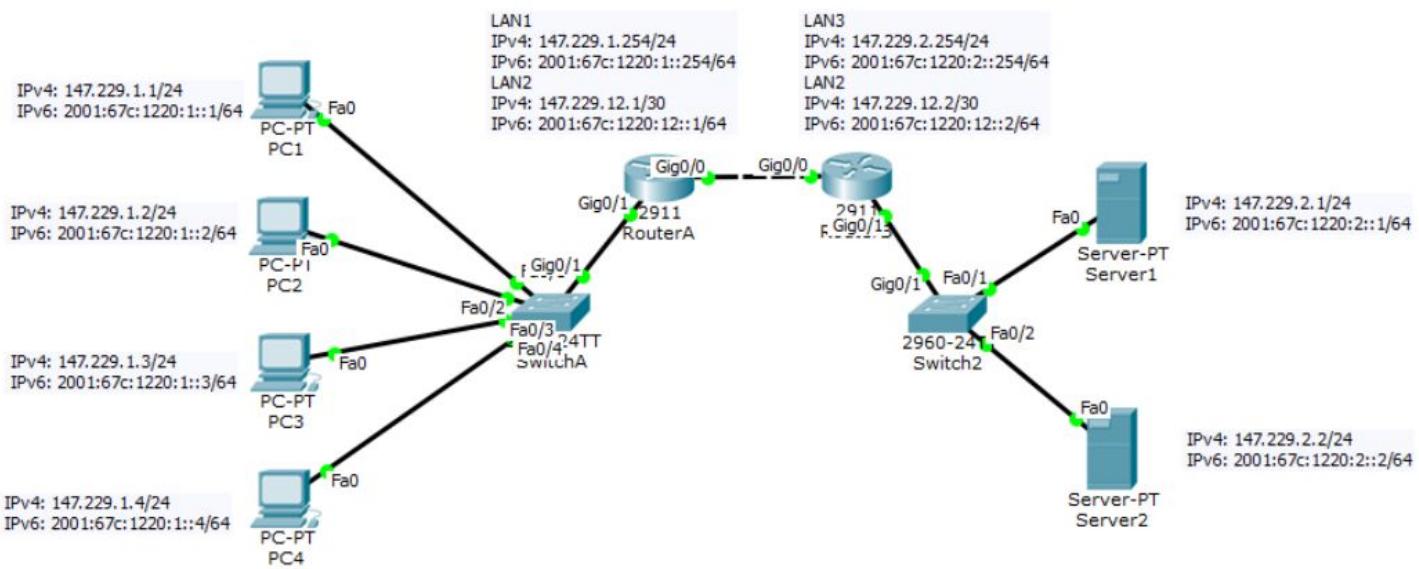
- nástroj zaměřený na automatickou analýzu zabezpečení kryptografických protokolů
- zapíšeme vlastnosti - specifikaci protokolu, vytvoří se simulace a verifikuje vlastnosti
- ověří důvěrnost a autenticitu a nalezne možné vektory útoku
- Princip:
  - nadefinuje se systém a komunikace
  - spustí se simulace
  - hledá všechny dosažitelné stavy, které by mohly mít nežádoucí vliv - na výstupu se někde objeví citlivá data
  - vrátí výsledky (nalezeno, nenalezeno, nevím)

## Verifikace konfigurace

- 0. získání topologie sítě (vím, nebo SNMP, NETCONF)
- 1. vytvoří se požadavky:
  - můžou se použít již vytvořeny v knihovně - hlídají i integritu sítě (pravidla)
    - unikátní ip, existující cíle / zdroje....
  - musí být splněny požadavky na konektivitu, bezpečnost, spolehlivost a výkonnost
- 2. vytvoří se konfigurace na základě požadavků
  - **může se použít nějaký vhodný jazyk, který ji pak vygeneruje**
  - už běžící síť se taky mění, můžou i vznikat nové
- 3. automatická verifikace (proverif říká jen ano / ne, tohle ukáže kde je chyba)
- 4. nasazení například pomocí SNMP, NETCONF

## Otázky

2] subnetting ipv4 a ipv6 s autokonfiguraci (jako na pulsemestrálce)



3] djkstruv algoritmus (priklad)

