PA159 – Počítačové sítě a jejich aplikace

# Rekapitulace předpokládaných znalostí

**Počítačová síť** je skupina vzájemně propojených počítačů nebo zařízení pomocí komunikačních kanálů. Může se používat pro:

* *Komunikaci* (text, řeč, video atd.)
* *Sdílení* (HW, SW, soubory, data, informace)

## Počítačové sítě obecně

### Základní charakteristika:

* Doručení – systém musí doručit data na správné místo
* Přesnost – systém musí doručit data správně a neporušeně
* Včasnost – systém musí doručit data dostatečně rychle

**Ideální sítě:**

* **Transparentní pro uživatele/aplikace**
* **Nekonečná propustnost**
* **Žádné ztráty**
* **Žádné zpoždění a rozptyl**
* **Dodržování pořadí paketů**
* **Data nemůžou být porušena**

**Skutečné sítě:**

* **vnitřní struktura ovlivňuje doručení dat**
* **omezená propustnost**
* **ztráty dat (někdy)**
* **zpoždění a rozptyl (někdy)**
* **míchání pořadí paketů (někdy)**
* **data mohou být poškozena**

**Požadované vlastnosti**

* **Efektivita – (téměř) maximální využití dostupných zdrojů**
* **Spravedlivost (stejná priorita různých toků)**
* **Decentralizovaná správa**
* **Rychlá konvergence při adaptaci na nový stav**
* **Spolehlivost**
* **Řízení toku dat (aby nenastalo zahlcení sítě)**

### **Základní přístupy**

**Spojované sítě (stavové) – spojení nazývané *okruh* je nastaveno mezi dvěma zařízeními po celou dobu komunikace**

* **Stav – informace o *okruhu* je spravována sítí**
* **Jednoduchá implementace**
* **Minimální škálovatelnost**
* **Příkladem je např. telefonní systém (ústředny, spojovatelky atd.)**

**Nespojované sítě – pro přenos dat není použitá žádná konkrétní cesta – data jsou rozkouskována na malé části = pakety a poslány přes síť. Pakety mohou být směrovány, kombinovány nebo rozdělovány na přijímacím zařízení (pakety jsou přečteny a poté složeny do původní podoby dat)**

* **Není třeba uchovávat stav sítě**
* **Složitější implementace**
* **Např. internet**

**Způsob komunikace musí být znám všem komunikujícím stranám. Formát komunikace a pořadí zpráv v komunikaci definuje tzv. Síťový protokol. Definuje také akce, které se provádí během přijímání či odesílání zpráv. Definuje:**

* **Syntaxi – struktura dat (pořadí v jakém jsou prezentována)**
* **Sémantiku – význam jednotlivých sekcí bitů**
* **Časování – kdy mají být data poslána a jak rychle mohou být poslána**

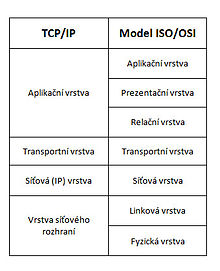
**Standardizace – definice norem popisující různé akce, aktivity, metody komunikace atd. (nejen v IT). Standardy by měly zaručovat kvalitu, bezpečnost, kompatibilitu, interoperabilitu a přenositelnost.**

**Druhy standardů:**

* **De Facto – někdo něco udělá a následně je to přijato komunitou**
* **De jure – standardy jsou určeny vyšší autoritou, zpravidla zákonem, IT organizacemi (ISO, ANSI, …).**

## **Síťové modely**

**ISO/OSI model:**sedmivrstvý model navržený organizací OSI za účelem dosažení kompatibility a interoperability komunikačních systému vyvinutých různými výrobci

* Každá vrstva je zodpovědná za určitou funkcionalitou (přidává kontrolní informace k datům kvůli správnému zpracování požadavků)
* Každá vrstva komunikuje jen s přímými sousedními vrstvami
* Funkcionalita je izolována k příslušné vrstvě (pokud se vrstva změní, musí se přizpůsobit pouze sousední vrstvy)
* Z logického hlediska spolu komunikují pouze stejné vrstvy, fyzicky však vždy komunikace probíhá přes všechny vrstvy
* Vrstvy jsou pouze abstraktní, implementace je jiná

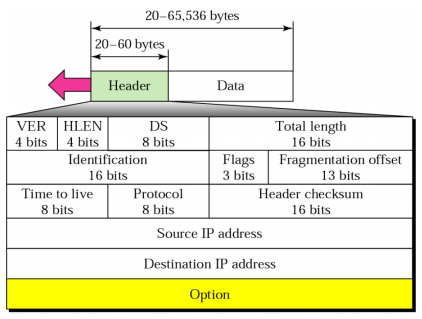
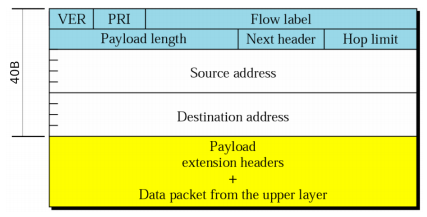
### Fyzická vrstva (L1)

* Zajišťuje end-to-end přenos bitů mezi odesílatelem a příjemcem
* Poskytuje funkcionalitu pro interakci s přenosovým médiem
  + Kontroluje počet odeslaných bitů za sekundu = **Bit-Rate Control**
  + Kontroluje synchronizaci přenosu mezi odesílatelem a příjemncem = **Bit Synchronization**
  + Rozděluje přenosové (fyzické) médium na logické kanály, které přenášejí data současně z různých zdrojů pro dosažení lepší efektivity = **Multiplexing**
    - Analogové signály: FDM, WDM; digitální signály: TDM
  + Umožňuje **přepínání okruhů**
* Poskytuje služby pro Vrstvu datových spojů
  + Vrstva datových spojů posílá/získává data fyzické vrstvě ve formě nul a jedniček uspořádaných do rámců
  + Fyzická vrstva transformuje proud bitů (z rámců) na signály, které šíří do přenosového média = **Bit-to-Signal Transformation**
* Data jsou přenášena ve formě elektromagnetických **signálů**
* Data, která musí být přenesena (nuly a jedničky), jsou digitální (binární)
* Signály šířené po přenosovém médiu jsou analogické nebo digitální
  + **Drátové médium** – fyzický vodič z jednoho zařízení do druhého, vhodné pro analogový i digitální přenos (koaxiální kabel, kroucená dvoulinka, optický kabel)
  + **Bezdrátové médium** – elektromagnetické vlnění (mikrovlny, radiové vlny, infračervené signály), vhodné pouze pro analogový přenos (vzduch, vakuum, voda)

### Vrstva datového spoje (L2)

* Přijímá pakety (ze síťové vrstvy) a převádí je na rámce (přidáváním hlaviček a patiček) = **Framing**
* Spolu s vrstvou L1 zajišťuje přenos rámců mezi komunikujícími zařízeními propojenými přenosovým médiem, např. LAN
  + **Adresování** entit fyzické vrstvy pomocí MAC adres
  + Rámce obsahují MAC adresu zdroje a cíle
  + MAC adresy jsou přiděleny výrobcem síťového zařízení
* Zajišťuje spolehlivost přenosu = **Detekce a oprava chyb**
  + Redundance: Odesílatel přidá navíc bity, jejichž hodnota je výsledek určité funkce na přenášených datech. Příjemce poté vypočítá stejnou funkci. Pokud se výsledek liší, podařilo se detekovat chybu a příjemce se ji pokusí opravit.
  + Pokud je chyba neopravitelná (nebo používáme jen detekci chyb bez oprav), příjemce požádá odesílatele o opětovné odeslání dat = *Automatic Request for Retransmition* (**ARQ**) – vhodné pro málo ztrátová přenosová média: sudá/lichá parita, CRC, …
  + *Forward Error Correction* (**FEC**) – vhodné pro ztrátová média (obzvláště když dochází k většímu zpoždění přenosu nebo pro multimédia)
* Zajišťuje **řízení toku dat**za účelem vyvarování se přetížení
* Řídí přístup více zařízení ke sdíleným médiím = **Medium Access Control**
  + Cíl: eliminace kolizí způsobených souběžným přenosem
  + Protokoly s neřízeným přístupem: Aloha, CSMA/CD, CSMA/CA
  + Protokoly s řízeným přístupem:
    - stanice smí vysílat pouze když získá svolení
    - rezervace, vyzývání (polling), předávání příznaku
  + Protokoly multiplexově orientovaného přístupu: FDMA, TDMA, …
* Umožňuje budovat **lokální sítě** (LAN – local area network) se systematickou tolologií: sběrnice, kruh, hvězda, strom atd.
  + Rozsáhlejší sítě jsou složeny ze vzájemně propojených jednoduchých topologií (LAN sítí)
  + Propojování zařízení: **bridge** = transparentní síťové spojení (všechen síťový provoz jde přes bridge), které odděluje sdílená média; **switch** = multi-port bridge
  + **Backward Learning Algorithm**: bridge se „učí“ adresy (MAC) jednotlivých síťových uzlů tím, že poslouchá média; rámce jsou poté přepínány na základě adresy odesílatele
* Je možné vytvořit sítě se smyčkami (cykly) => využívání **Distributed Spanning Tree Algorithm**
  + Cíl: odstranit cykly při zasílaní dat mezi jednotlivými můstky (bridges)
  + Všechny můstky periodicky posílají svým sousedům informaci o své vzdálenosti od kořene a upravují svoji definici nejlepší cesty
  + Jakmile bridge obdrží zprávu, přizpůsobí svou nejlepší cestu:
    - Preferuje se kořen s nižší adresou
    - Preferuje se cesta s nižší cenou

### Síťová vrstva (L3)

* Poskytuje služby Transportní vrstvě (L4): L3 obdrží segmenty z L4 a transformuje je na pakety = **Packetizing**
* Spolu s L2 zajišťuje přenos paketů mezi komunikačními uzly
  + IP protokol: datagramový přístup k přepínání paketů; komunikace bez připojení – **routování** posílaných paketů – proces výběru cest v síti mezi dvěma uzly
  + Komunikace mezi bezprostředními uzly (routery) – **host-to-gost delivery**
  + Komunikace mezi různými LAN sítěmi
  + Nespolehlivá služba (nazývana **best-effort**)
  + Doplněna sadou protokolů (ICMP, ARP, RARP, IGMP) – používáno pro nestandartní situace, pro distribuci informací potřebných pro správné směrování, pro rozlišování MAC adres atd.
* Umožňuje jedinečnou identifikaci (**adressing**) každého zařízení v celém Internetu
  + L3 entity jsou identifikovány IP adresami – unikátní napříč celou sítí
  + Pakety obsahují adresy zdrojové a cílové komunikující entity
  + IPv4 adresy (32 bitů, 4 skupiny čísel v desítkové soustavě oddělené tečkami)
  + IPv6 adresy (128 bitů, 8 skupin čísel v šestnáctkové soustavě oddělených dvojtečkou
  + Typy adres IPv6:
    - **Unicast adresa** – identifikace jednoho zařízení
    - **Multicast adresa** – identifikace skupiny zařízení, které požaduje data
    - **Anycast adresa** – stejné jako multicast, ale data jsou doručena pouze jednomu členu ze skupiny
* L3 logicky spojuje nezávislé LAN sítě = **Internetworking**
  + Takovým spojením pak vzniká internetwork, zkráceně *internet*
* IPv4 datagram:
  + **Verze** (VER) – verze IP protokolu
  + **Délka hlavičky** (HLEN) – délka hlavičky IP datagramu
  + **Differentiated services** (DS) or **Type of service** (TOS) – definuje třídu datagramu
  + **Celková délka** – délka celého datagramu, maximálně 216 – 1 B
  + **Identifikace, Flagy, Offset** – pole používaná pro fragmentaci
  + **TTL** (Time to live) – ochrana proti zacyklení. Každý směrovač zmenší tuto hodnotu o jedničku. Jakmile je TTL = 0, datagram se zahodí.
  + **Protokol** – určuje, kterému protokolu vyšší vrstvy se mají data předat při doručení (IANA, např. 1 = ICMP, 6 = TCP, 17 = UDP atd.)
  + **Kontrolní součet hlavičky** – slouží k ověření, zda nedošlo k poškození.
  + **Zdroj/cíl** – IP adresa zdroje a cíle
  + **Volby** – rozšiřující informace či požadavky
  + **Data** – další zapouzdřené protokoly
* **Kontrolní zprávy** – ICMP protokol:
  + Podpůrný protokol pro IP protokol
  + Poskytuje informace o chybách vzniklých při doručení dat
  + Poskytuje základní informace o stavu sítě
  + Zprávy (data přenášená v paketech):
    - *Destination unreachable* – port, protokol, host, síť
    - *Time exceeded* – TTL expirace nebo chybějící části zprávy
    - *Echo request/reply* – žádost o odpověď
* IPv6:
  + Větší adresovací prostor – 128 bitů, až 2128 unikátních adres
  + Jendodušší 40 B hlavička obsahuje většinu důležitých informací
    - S možností rozšíření pomocí *extension headers*
  + Podpora real-time přenosů
  + Podpora mobility – pomocí tzv. *home agents*
  + Podpora autokonfigurace zařízení – stavová a bezstavová autokonfigurace
* IPv6 datagram:
  + Kontrolní hlavičkový součet odstraněn
  + Informace o volbách a fragmentaci dostupné pomocí rozšiřujících hlaviček
  + **Verze** (VER) – verze IP adresy
  + **Priorita** (PRI), také **Traffic class**
  + **Flow label** – Značka toku, poskytuje speciální řízení pro částečný tok dat
  + **Délka dat** (Payload length) – celková délka IP datagramu, nezahrnuje hlavičku
  + **Další hlavička** (Next header) – definuje hlavičku, která následuje ihned za základní hlavičkou (base header)
  + **Hop limit –** stejné jako TTL v IPv4
  + **Adresa zdroje/cíle** – IPv6 adresa zdroje a cíle
* ICMPv6 – založeno na mechanismu ICMPv4, navíc zahrnuje funkcionalitu ARP (Adress Resolution) a IGMP (skupinová správa pro multicast)

#### Routing (Směrování)

* Proces hledání cesty v síti mezi dvěma komunikačními uzly
  + Cesta musí být optimální
  + Cíl: doručit data k příjemci
* Proces je ovlivněn statickými faktory (topologie sítě) a dynamickými faktory (zátěž sítě)
* **Routovací tabulka** představuje **lokální pohled** na topologii sítě
  + je obtížné dosáhnout globální znalosti o topologii sítě
  + I když by bylo dosaženo globální znalosti sítě, rozdíly mezi lokálním a globálním pohledem na síť mohou vést k cyklům nebo oscilacím při síťové komunikaci
* Proto routování (obvykle) neřeší celou cestu paketu
  + Router zajišťuje jen jednotlivé kroky
  + Hop-by-hop princip: komu má být konkrétní packet předán, abychom se co nejvíce přiblížili k příjemci
  + Další router se poté rozhodne co dál provést s přijatým packetem

Routování spadá do kategorie grafových problémů. Síť může představovat graf, kde jsou uzly reprezentovány jednotlivými routery (identifikovány IP adresami), hrany reprezentují propojení směrovačů a hodnoty hran jsou cena cesty (komunikace). Cílem je nalezení cesty mezi dvěma uzly v síti s co nejmenší cenou.

* **Statické** routování: nepřizpůsobuje se, je vhodné pro menší neměnící se sítě
* **Dynamické** routování: reaguje na změny sítě, využívá komplexnější algoritmy
  + Centralizované – jeden uzel řídí zbytek
  + Izolované – každý uzel pracuje sám
  + Distribuované – uzly spolupracují
* Požadované vlastnosti směrovacího algoritmu:
  + Přesnost
  + Jednoduchost
  + Efektivita a škálovatelnost
  + Robustnost a stabilita
  + Férovost
  + Optimálnost
* **Distance Vector (DV)** algoritmy
  + Číslo vektoru vzdálenosti – metrika indikující počet skoků v síti
  + Sousední směrovače si periodicky mění celé své routovací tabulky
  + Router si na základě přijatých updatů aktualizuje své informace a zvyšuje si Distance vektor number (používá se Bellman-Fordův algoritmus)
* **Link State (LS)** algoritmy
  + Routery si periodicky vyměňují informace o stavů svých přímých propojení
  + Routery si udržují kompletní informace o topologii sítě – každý router ví o všech dalších routerech v síti
  + Využívá se Dijsktrova algoritmu pro vypočítání nejkratší cesty mezi uzly
* Distance Vector – RIP protokol
  + RIPv1 a RIPv2
  + Sítě jsou identifikovány pomocí CIDR mechanismu
  + Metriky: počet skoků (hops)
    - Přenos paketu mezi dvěma sousedními směrovači = 1 hop
    - Infinity = 16 => RIP protokol nemůže být použít pro sítě s minimálním počtem skoků mezi dvěma routery > 15
  + Routery posílají informace periodicky každých 30 s
    - Timeout je 180 s
* Link Statce – OSPF protokol
  + Metriky: cena – číslo od 1 do 65535, které je přiřazeno každému síťovému rozhraní jednotlivých routerů
  + Čím je cena nižší, tím je daná cesta lepší.
  + Standardně je každému rozhraní přiřazena cena odpovídající propustnosti linky
  + Cena = 109/šířka pásma (v jednotkách bps)
  + Cena může být manuálně upravena
* **Link State:**
  + Složitost – změna ceny linky se propaguje na všechny uzly, O(nE) zpráv
  + Rychlost konvergence – algoritmus O(n2), zasílá O(nE) zpráv, trpí na oscilace
  + Robustnost – špatný výpočet se šíří jen sousedům
  + Použití – menší i rozsáhlé sítě
* **Distance Vector:**
  + Složitost – změna ceny linky se propaguje jen nejbližšímu sousedovi a těm, které to ovlivní
  + Rychlost konvergence – count-to-infinity problém (směrovací cykly), může být pomalejší než LS
  + Robustnost – nesprávný výpočet je šířen sítí
  + Použití – menší sítě

Dělení internetu na **Autonomní systémy**

* Cílem je rozdělení internetu na autonomní systémy, abychom zajistili a zmírnili přetížení routerů
* Autonomous System Number (ASN) – 16ti bitové ID
* Autonomní systémy odpovídají administrativním doménám (např. CESNET, PASNET apod.)
* Typy podle připojení k Internetu:
  + Stub
  + Multihomed
  + Transit
* Oddělené směrování:
  + Interní – v rámci AS, protokoly: RIP, OSPF, (E)IGRP, IS-IS. Pod plnou kontrolou administrátora. Cílem je především výkon.
  + Externí – mezi AS, protokoly: BGP-4, cílem je především podpora definovaných pravidel a škálovatelnosti

### IP multicast

* Skupinová komunikace v síti (globální, nejen přes LAN)
* Přes každý síťový spoj putuje pouze jediná kopie dat
* Jedná se o funkcionalitu sítě (hop-by-hop service)
* Nespolehlivé doručování (best effort, UDP, skupinové adresování)
* Omezeno pomocí TTL
* Skupina je identifikována pomocí multicast IP adresy:
  + IPv4: třída D (244.0.0.0 – 239.255.255.255)
  + IPv6: prefix ff00::/8

#### Sourced Based Tree

* Aktivita shora od zakládajícího
* Periodický broadcast
* Ořezávání větví bez členů
* Omezení šířky TTL
* Protokoly: DVMRP (RIP), MOSPF (OSPF), PIM-DM

#### Shared Tree (Core Based Tree)

* Ustanovené jádro – body setkání (MP)
* Zájemce o skupinu kontaktuje MP
* Aktivita zdola od příjemce
* Redukce broadcastu -> lepší škálování
* Závislost na dostupnosti jádra
* Protokoly: CBT, PIM-SM (protokolově nezávislé)

### Transportní vrstva (L4)

Poskytuje služby Aplikační vrstvě.

* Získává data od odesílající aplikace a transformuje je do segmentů (přidáním transportní hlavičky) = **paketizace**

Ve spolupráci s vrstvou L3 zajišťuje doručení dat (segmentů) mezi komunikujícími aplikacemi/procesy

* Zajišťuje spolehlivost, poskytuje logický kanál
* Proces-to-process delivery
* **Connection-oriented services** – před přenosem je vytvořeno spojení, pakety jsou očíslovány a jejich doručení/nedoručení je explicitně potvrzeno
* **Connection-less services** – pakety jsou poslány cílové aplikaci bez toho, aby bylo vytvořeno spojení, pakety nejsou očíslovány a mohou se ztratit, opozdit, doručit ve špatném pořadí atd.

Adresy entit z vrstvy L4 se nazývají **porty**.

* Pakety obsahují zdrojová a cílová čísla portů – 16 bitová čísla (0 – 65535), které identifikují odesílající a přijímající aplikaci
* Aplikace je identifikována unikátně pomocí dvojice IP adresa:port

Poskytuje spolehlivost spojení, pokud je třeba

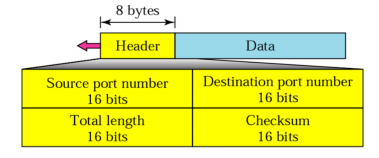
* Kontrola toku dat a kontrola chyb
* Poskytováno díky principu node-to-node (od uzlu k uzlu), poskytováno nižšími vrstvami
* Zajišťuje spolehlivost díky best-effort service (IP)

Kontrola přetížení a kvality služeb (Qos – Quality of Service)

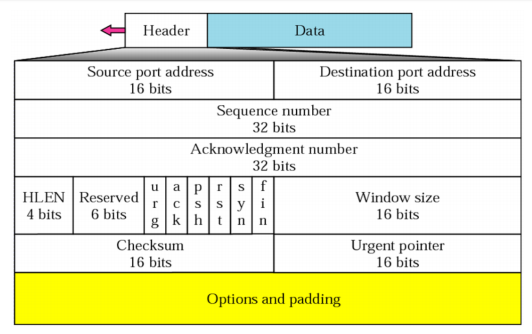
Nejnižší vrstva poskytuje tzv. end-to-end služby

* Hlavičky jsou generovány odesílatelem a interpretovány pouze příjemcem
* Router vidí data vrstvy L4 jako payload („náklad“, to co si pakety nesou s sebou) přenášených paketů

### User Datagram Protocol (UDP)

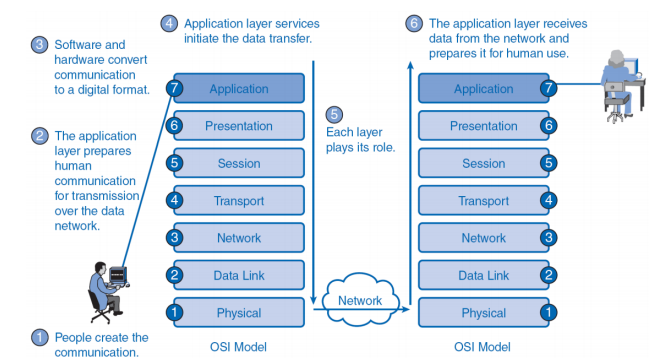
* Nejjednodušší transportní porotkol poskytující **connection-less** a **nespolehlivé** (=best-effort) služby
* Obohacuje služby vrstvy L3 jenom o process-to-process komunikaci a jednoduchou kontrolu chyb
* Pokud musí být zajištěna spolehlivost, musí být poskytnuta aplikací
* Jednoduchost, minimální režijní náklady
  + Není vytvářeno připojení (=> žádné zpoždění na začátku přenosu)
  + Žádná údržba stavu v uzlech, které spolu komunikují
  + Jednodušší a menší hlavička
* Aplikace:
  + Procesy, které vyžadují jenom jednoduché požadavky – reply communication (DNS)
  + Procesy/protokoly s interním tokem a kontrolou chyb (TFTP)
  + Real time přenosy
  + Multicast přenosy
* UDP Header:
  + 
  + **Zdrojový port** – identifikace odesílající služby/aplikace
  + **Cílový port** – identifikace příjemce
  + **Délka** – celková délka hlavičky UDP paketu (hlavička + data)
  + **Checksum** – kontrolní součet (hlavička + data)

### Transmission Control Protocol (TCP)

* Transportní protokol poskytující **connection-oriented** a plně **spolehlivou** službu
  + Pokud je to možné, data jsou odeslána a přijata ve správném pořadí
  + V porovnání s UDP je TCP orientováno spíše na proud bytů (UDP pracuje s bloky dat)
* Mezi odesílatelem a příjemcem musí být vytvořeno spojení
  + Před samotnou komunikací zajišťuje takzvaný třícestný handshake (z anglického „potřesení ruky“) výměnu všech potřebných informací
  + Spojení je rozlišitelné pouze na koncových uzlech (end-to-end service) – routery si nejsou vědomy spojení
  + Vytvořené spojení může být použito pro full-duplex komunikaci
  + Podporována jsou jenom point-to-point spojení – komunikace mezi více než dvěma uzly není podporována
* Multiplexing/demultiplexing a kontrola chyb jsou v rámci UDP to stejné
* TCP Header:
  + 
  + **Zdrojový port** – identifikátor odesílatele (služby/aplikace)
  + **Cílový port** – identifikátor příjemce (služby/aplikace)
  + **Číslo posloupnosti (Sequence number)** – číslo přiřazené prvnímu bajtu dat obsažených v segmentu
  + **Acknowledgement number** – číslo, které příjemce očekává v dalším přijatém segmentu
  + **HLEN** – celková délka TCP hlavičky (4B slova)
  + **Control** – 6 kontrolních bitů, které identifikují různé kontrolní informace
  + 
  + **Velikost okna** – velikost okna, kterou musí opačná strana spravovat (používáno pro službu kontroly toku)
  + **Checksum (kontrolní součet)** – kontrolní součet TCP segmentu (header + data)
  + **Urgent pointer** – pokud má segment naléhavá data
* TCP kontroluje množství posílaných dat těmito způsoby:
  + Chrání příjemce před přetížením = **Flow Control**
  + Chrání síť před přetížením = **Congestion Control**
* Množství dat, které je povoleno poslat do sítě, je definováno:
  + Velikostí okna příjmence
  + Velikostí okna přetížení – na straně odesílatele
* Množství dat zasílaných do sítě je limitováno **nižším** z obou parametrů

### Aplikační vrstva (L7)

* Aplikace – hlavní důvod, proč existují počítačové sítě
* L7 poskytuje služby uživatelům: e-mail, WWW, DNS, http, FTP, RTP, …
* Skládá se ze síťových aplikací/programů a aplikačních protokolů
  + Aplikační protokoly (http, SMTP atd.) jsou části síťových aplikací
  + Nejsou samostatnými aplikacemi
  + Definují syntaxi, sémantiku a časování zasílání zpráv



* Klasifikace podle využívaného komunikačního modelu
  + Client-Server model (Thin vs. Fat client)
  + Peer-to-peer model
* Klasifikace podle způsobu přístupu k informacím
  + Pull model – přenos dat je iniciován klientem
  + Push model – přenos dat je iniciován serverem
* Klasifikace podle nároků na počítačovou síť
  + Aplikace s nízkými nároky na počítačovou síť
  + Aplikace s vysokými požadavky na počítačovou síť

# IPv6 – pokročila funkcionalita

## Adresování

* Zkratky IPv6 adres
  + Vynechání nul zleva
  + Vynechání bloků obsahujících samé nuly
* Hierarchie IPv6 adres:

|  |  |  |
| --- | --- | --- |
| **n bitů** | **64-n bitů** | **64 bitů** |
| Global routing prefix | Subnet address | Interface address |

* Global routing prefix = globální směrofací prefix
  + Global routing prefix = globální směrovací prefix
  + Subnet address – adresa podsítě
    - Obvykle je 16 bitů dlouhá => globální směrovací prefix má 48 bitů
    - Prvních 16 bitů obsahuje hodnotu 2001 (hexadecimální forma)
    - Dalších 16 bitů je přiřazeno RIR (Regional Internet Registry)
    - Dalších 16 bitů je přiřazeno LIR (Local Internet Registry)
  + Interface address – adresa rozhraní
* IPv6 adresy jsou rozděleny do tříd (třídy A, B, C, D, E neexistují)
* IPv6 sítě jsou definovány pomocí CIDR notace (jako v případě IPv4), např. FDEC::BBFF:0:FFFF/60. Neoddělujeme adresu sítě a podsítě
* Typy IPv6 adres
  + Unicast adresa – stejné jako u IPv4, jednomu ze sítě
  + Multicast adresa – stejné jako u IPv4, všem ze skupiny
  + Anycast adresa – novinka v IPv6, identifikuje skupinu zařízení/hostů stejně jako v případě multicastu, ale data jsou doručována jen jednomu členu skupiny (tomu „nejbližšímu“)
  + IPv4 broadcast adresy se v IPv6 nepoužívají (jsou nahrazeny speciální multicast adresou FF02::1 – všechny uzly v konkrétní LAN síti

## Path MTU Discovery

* MTU = Maximum Transmission Unit
* Pouze zdrojová zařízení musí rozhodovat o správné velikosti fragmentů (routery nemají možnost fragmentovat jednotlivé datagramy, to můžou dělat jen koncové uzly)
* Pokud je datagram pro router příliš velký, musí ho router zahodit a poslat zpětnou vazbu do zdroje (ICMPv6 Packet Too Big)
* Path MTU Discovery je technika, která se používá pro rozhodování o tom, jaká by měla být použita velikost fragmentu
* Používá techniku zpětné vazby, která je prováděna pomocí ICMPv6 Packet Too Big zpráv
  + Zdrojový uzel pošle datagram, který má MTU svého lokálního fyzického spojení (reprezentováno horní hranicí MTU)
  + Pokud se podaří přenos bez chyby, hodnota MTU může být použita i pro další datagramy posílané do příslušné cílové lokace
  + Pokud se při přenosu vrátí zpět jakákoliv zpráva Packet Too Big, provede se pokus o přenos znovu spolu s využitím menší velikosti datagramu

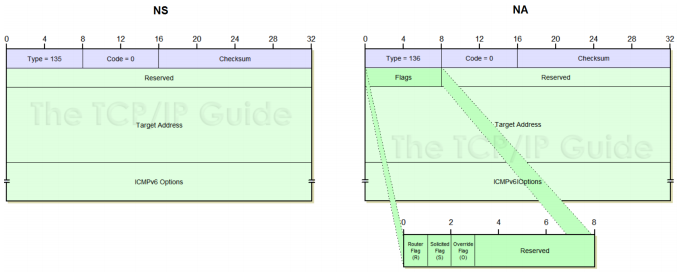
## Neighbor Discovery Protocol (NDP)

NDP je součást ICMPv6. Využívá pět ICMP zpráv:

* Neighbor Solicitation (NS) – Získává L2 adresy, DAD, NUD. Solicitation = žádost
* Neighbor Advertisement (NA) – odpovídá na NS, informuje o změnách L2 adres
* Router Solicitation (RS) – host žádá router o informaci o autokonfiguraci
* Router Advertisement (RA) - obsahuje informace o autokonfiguraci
* ICMP Redirect – pokud výchozí brána pošle Redirect zprávu (přesměrování) zdrojovému hostiteli, hostitel může změnit hop router za účelem posunutí paketů vpřed.

### Rozlišování L2 adres

* Založeno na NS (challange) a NA (odpověď) ICMP zprávách
* Obecný multicastový prefix je předem definován
  + Např. FF02:0:0:0:0:1:FF00::/104
* Uzel, který hledá L2 adresu, si vezme posledních 24 bitů IP adresy, jejíž L2 adresu hledá a spojí ji s prefixem
  + Např. při hledání L2 adresy IP adresy 2AC0:56:A319:15:022A:FFF:FE32:5ED1 dostaneme FF02:0:0:0:0:1:FF32:5ED1
  + Cílová adresa je multicastovou adresou
  + Posledních 24 bitů zajišťuje, že multicastová skupina bude obsahovat jen několik uzlů (typicky 1 nebo 0)
* NS zpráva je poslána takové multicastové adrese, která obsahuje IPv6 adresu, která bude rozhodnuta a L2 adresu odesílajícího uzlu.
* Soused musí naslouchat takovým zprávám ve své multicast skupině
* Jakmile uzel (jehož IPv6 adresa bude rozhodnuta), který patři příslušné multicastové skupině, přijme NS zprávu, odpoví pomocí NA zprávy
* Odpověď obsahuje
  + Všechny IPv6 a L2 adresy, které uzel má
  + Atribut: R (Router) / S (byla vyžádána NA zpráva?) / O (Měla by nová informace přepsat informace, které byly na uzlu uloženy dříve?)

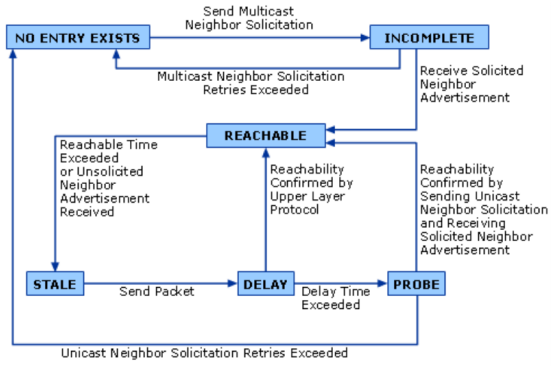


### Duplicate Address Detection (DAD) – Detekce duplicitních adres

* Používáno během procesu autokonfigurace
* Hostitel pošle NS zprávu se svou vlastní (multicastovou) adresou jako cílovou adresou a požádá o link-adresu v adrese zdrojového paketu
  + Zdrojová adresa je nejprve nastavena jako nespecifikováná „::“, tzn. samé nuly
* Hostitel pošle zprávu se svojí vlastní adresou, pokud se nějaký uzel ozve (zprávou NA), nachází se v síti duplicita

### Neighbor Unreachability Detection (NUD) – Detekce nedosažitelnosti souseda

* Uzel pravidelně kontroluje dostupnost svých sousedů
* Dva přístupy:
  + Vysokoúrovňový protokol (např. TCP) informuje IPv6, že komunikace proběhla a tím pádem je uzel přístupný
  + V opačném případě musí IPv6 provádět takovou detekci samo
* Adresa, která je uložená v cache, se může nacházet v následujících stavech
  + Incompete – rozlišení adresy je ještě v procesu a čeká se na další odpověď nebo a timeout (NS zpráva už byla poslána, ale příslušná NA zpráva ještě nebyla přijata)
  + Reachable – soused je aktuálně dostupný (bylo obdrženo pozitivní potvrzení v rámci poslední ReachableTime)
  + Stale – uplynulo už více než ReachableTime milisekund od doby, kdy přišlo poslední pozitivní potvrzení
  + Delay – sousedův ReachableTime vypršel, prokol vyšší vrstvy může potvrdit dosažitelnost v rámci specifického času
  + Probe – aktivní pokusy o potvrzení dostupnosti

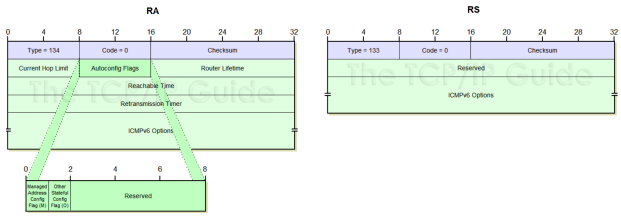


### Autokonfigurace

* Manuální konfigurace je neproveditelná v rámci velkých sítí nebo v rámci sítí s velkým množstvím zařízení
* Stavová konfigurace – jako DHCP v IPv4 (zde se nazývá DHCPv6)
* Bezstavová konfigurace – nový typ autofonfigurace, který budeme studovat
* Oba typy můžeme kombinovat:
  + Bezstavová autokonfigurace může být použita na generování IPv6 adress a také
  + Stavová autokonfigurace může být použita k získání dodatečných parametrů
* Bezstavová autokonfigurace předpokládá, že v rámci sítě existují chytré routery, které ví vše potřebné
* V náhodných časových intervalech každý router informuje všechny uzly (multicast) o aktuální konfiguraci (Router Advertisement)
* Nový uzel může počkat na RA (Router Advertisement), nebo si o něj explicitně požádat (Router Solicitation) – RA je pak posláno jako odpověď na RS na unicast adresu
* Aby host vygeneroval IP adresu, používá kombinaci lokálních informací (jako např. svou MAC adresu nebo náhodně vybrané ID) a informace, které host přijal od routerů

Kroky, které zařízení provede, když využívá bezstavovou autokonfiguraci

1. Link-Local Address Generation – zařízení vygeneruje link-local adresu (tzv. orientační adresu): 1111 1110 10 jako prvních 10 bitů (prefix FE80) následovaných 54 nulami, následovaných 64 bitovým identifikátorem rozhraní (MAC adresa nebo náhodně vybrané ID)
2. Test unikátnosti Link-Local Address – je třeba zajistit, aby adresa, kterou uzel vygeneroval, nebyla již použita někde v lokální síti (to je nepravděpodobné, pokud link-local adresa pochází z MAC adresy, ale více pravděpodobné, pokud je link-adresa založena na náhodně generovaném ID).
3. Přidělení Link-Local Adresy – za předpokladu, že test unikátnosti prošel, zařízení přiřadí link-local adresu do svého IP rozhraní
4. Router Contact – uzel naslouchá na RA zprávy nebo posílá RS zprávy, aby získal více informaci o pokračování v autokonfiguraci
5. Router Direction – router instruuje uzel o tom, jak rozhodnout svou adresu v celém Internetu, nebo informuje, že stavová autokonfigurace se již používá a poskytne ji spolu s IP adresou DHCP serveru
6. Global Address Configuration – za předpokladu, že se bezstavová konfigurace již používá, host si sám nastaví unikátní Internetovou adresu: utvořenou ze síťového prefixu je poskytnuta routerem hostiteli, dále kombinována s identifikátorem zařízení



* Autokonfigurační příznaky: M = 1: Používá stavovou metodu pro konfiguraci adres (DHCPv6), O = 1: Používá stavovou metodu pro jiné informace než adresy
* Router lifetime – Jak dlouho by měl být daný router používán jako defaultní router (0 = router by neměl být používán jako defaultní
* Reachable time – čas dosažitelnosti – jak dlouho by měl host považovat souseda za dosažitelného poté, co obdržel potvrzení dosažitelnosti
* Retransmission timer – počet milisekund, které by měl host čekat před opětovném odesíláním
* ICMPv6 volby – RA zprávy můžou obsahovat tři možné volby:
  + Zdrojová L2 adresa
  + MTU
  + Prefix informace

## Mobility support – podpora mobility

Použité adresy:

* Domácí adresy – globální unicastové adresy, které se nemění (persistentní), přes které je mobilní uzel vždy přístupný
* Care-of adresy – globální unicastové adresy pro mobilní uzly, které jsou v cizí síti
* Spojení domácí adresy a care-of adresy se nazývá *binding*

Komunikující uzly:

* Mobilní uzel (Mobile Node – MN) – mobilní zařízení
* Korespondenční uzel (Correspondent Node – CN) – peer uzel/uzly, se kterým MN komunikuje
* Home Agents (HA) – router(y) v domácí síti, přes které je MN vždy dostupné
  + Vazby procesů, care-of adresy procesů a jejich změny
  + Údržba cache bindingu a seznam home agentů
  + Přeposílá pakety (tunnels) na care-of adresu
  + Podporuje HA Address Discovery

Když je MN doma, přijímá pakety přes obvyklý IP směrovací mechanismus a chová se jako jakýkoliv jiný host. Když je MN mimo domácí síť, sám si registruje svoje care-of adresy spolu s jejich HA.

Existují dva způsoby, jak mezi s sebou můžou MN a CN komunikovat:

* Dvojsměrné tunelování (Bidirectional tunneling) – pakety z CN jsou poslány na HA, který je zapouzdří a pošle je dál na care-of adresu MN (a naopak)
* Optimalizace směrování (Route optimization) – přímá komunikace MN a CN
  + MN si musí zaregistrovat care-of adresu spolu s CN
  + Navázání spojení musí být autorizováno pomocí Return Routability Procedure

### Return Routability Procedure (RRP)

* Podpora mobility není bezpečná – risk situace *man-in-the-middle* (něco jako odposlouchávání během přenosu)
* MN musí dokázat CN, že vlastní home adresu a i care-of adresu
  + Nesdílí ale žádný klíč
* RRP umožňuje CN získat rozumné ujištění, že MN zpráva je adresovatelná přes svou záskanou care-of adresu a stejně tak přes svou home adresu – jen když je to úspěšně prokázáno, může dojít k optimalizaci cesty v síti

1. MN pošle Home Test Init (HoTI) zprávu pomocí HN na CN. Tato zpráva nese Home Init Cookie, takže CN si zjistí home adresu MN.
2. MN pošle Care-of Test Init (CoTI) zprávu přímo na CN. Tato zpráva nese Care-of Init Cookie, Takže CN si zjistí care-of adresu MN.
3. CN odpoví na HoTI pomocí Home Test (HoT) zprávy, kterou pošle pomocí HA. Tato zpráva nese Home Init Cookie a Home Nonce Index. MN nyní může vygenerovat Home Keygen Token
4. CN odpoví na CoTI zprávu pomocí Care-of Test (CoT) zprávy, kterou pošle na care-of adresu MN. Tato zpráva obsahuje Care-of Init Cookie a Care-of Nonce Index. MN nyní může vygenerovat Care-of Keygen Token
5. MN i CN vypočítá 20-B Management Key, který se používá k zabezpečení Binding Update zprávy. Díky správnému Management Key MN dokázalo, že je dosažitelné jak přes home adresu, tak přes care-of adresu

## IPv6 bezpečnost

Standartní postupy síťové bezpečnosti zahrnují dvě „triády“, CIA a AAA:

* *Confidentiality –* důvěrnost. Uloženné nebo přenášeníí informace nesmí být přečteny nebo pozměněny neautorizovanou stranou
* *Integrity –* integrita. Jakákoliv změna přenášené nebo uložené informace může být detekována.
* *Availability –* dostupnost. Informace je přístupná autorizovaným uživatelům v jakýkoliv čas.
* *Authentication –* Autentifikace. Prokázání získané identity.
* *Authorization –* Autorizace. Ujištění se, že autentifikovaný (přihlášený) uživatel nebo skupina má práva na přístup k informaci, ke které se pokouší dostat.
* *Accounting –* Sbírání informací o využití zdrojů
* *+ Non-repudiation –* (repudiation = odmítnutí) Specifická akce (jako např. odesílání, přijímání nebo mazání informace) nemůže být zamítnuta žádnou ze zúčastněných stran.

Bezpečnostní požadavky na bezpečnost musí být zajištěny dvěma základními bezpečnostními prvky:

* Šifrování (Encryption) – k zajištění důvěryhodnosti
* Bezpečné kontrolní součty – k zajištění integrity
* Vhodné kombinace obou těchto prvků mohou být využity k zajištění komplexnějších služeb jako je autentifikace a non-repudiation
* Tyto bezpečnostní mechanismy jsou popsány v IPSec frameworku
  + IPv4: IPSec může být instalován samostatně
  + IPv6: IPSec je doporučená funkcionalita IPv6 stacku (balíku)

#### Prvky IPSec frameworku:

* Authenticatio protocol – AH (Authentication Header)
* Encryption protocol – ESP (Encapsulating Security Payload) header
* Definice využití kryptografických algoritmů pro šifrování a autentifikaci
* Definice bezpečnostních pravidel a bezpečnostních spojení mezi komunikujícími uzly
* Správa klíčů

#### Security Associations (SA)

* Sada bezpečnostních informací, které popisují příslušný druh zabezpečeného spojení mezi dvěma zařízeními
* Definováno třemi parametry: Security Parameter Index (32-bit ID), IP Destination Address (adresa zařízení, pro které je SA ustanoveno), Security Protocol Identifier (AH nebo ESP)
* Skládá se ze tří prvků: klíč + šifrovací nebo autentifikační mechanismus + dodatečné parametry pro algoritmus
* SA jsou jednosměrné dohody => poskytnout šifrovanou a autentifikovanou komunikaci mezi dvěma uzly, čtyři SA jsou nutné

#### Key Management – správa klíčů

Za účelem vytvoření SA, jednotlivé peery musí souhlasit s kryptografickým algoritmem a vyjednat si své klíče. Nicméně jednání o klíčích často probíhá na nezabezpečené síti. Následující protokoly byly navrhnuty pro automatizaci vyjednávání klíčů:

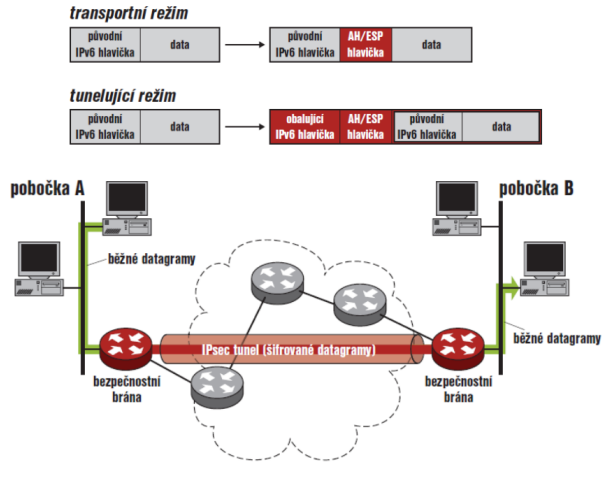
* V minulosti: ISAKMP, IKEv1 (Internet Key Exchange)
* Nyní: IKEv2 (Zjednodušené a opravené IKEv1)

1. Vytvoření zabezpečeného kanálu k vyjednávání o ochraně dat
2. Vytvoření/smazání crypto materiálu – vede k jednoduchému ISAKMP/IKE SA
3. Vytvoření zabezpečeného kanálu pro přenos dat – vede k páru IPSec SA.

### IPSec režimy přenosu

Režim přenosu:

* Režim transportu:
  + Protokol chrání zprávy, které prošly dolů směrem k IP z transportní vrstvy
  + Zpráva je zpracována pomocí AH/ESP a na začátek transportní hlavičky (UDP nebo TCP) jsou přidány příslušné hlavičky
  + IP hlavička je poté přidána na začátek
* Režim tunelu:
  + IPSec se používá k ochraně a ke kompletnímu zapouzdřeného IP datagramu poté, co je aplikována IP hlavička
  + IPSec hlavička se objeví na začátku původní IP hlavičky a nová IP hlavička se přidá na začátek IPSec hlavičky
  + Proto je celý původní IP datagram zabezpečen a zapouzdřen pomocí jiného IP datagramu

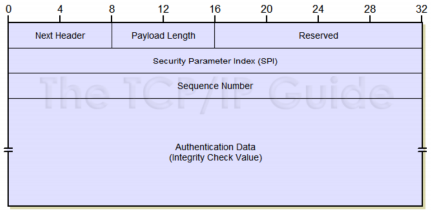


### Authentication Header a Encapsulating (Zapouzdření) Security Payload

Authentication Header (AH):

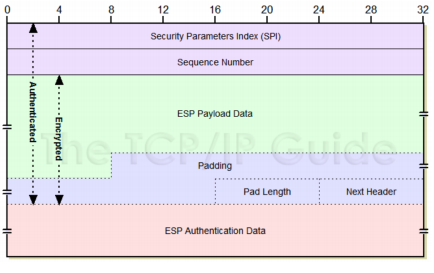
* Protokol, který nabízí autentifikaci celého datagramu nebo jeho části
* Umožňuje ověřit integritu zprávy, ale nezašifruje ji

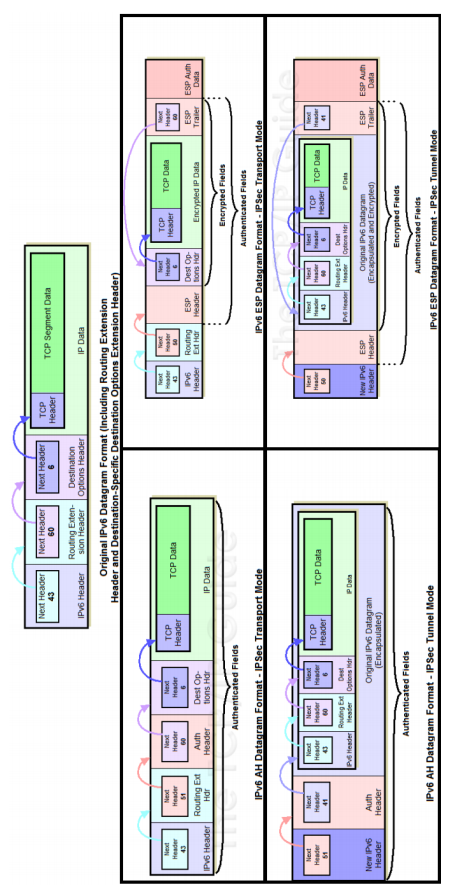
1. SA musí být nastaveno mezi dvěma komunikujícími zařízeními (jen zdroj a cíl ví, jak provést výpočet)
2. Na zdrojovém zařízení AH provede výpočet založený na hodnotách v datagramu a výsledek (nazývaný Integrity Check Value – ICV) dá do speciální hlavičky spolu s ostatními poli pro přenos
3. Cílové zařízení provede stejný výpočet s využitím klíče, který dvě zařízení sdílí, což mu umožní ihned vidět, zdali v původní hlavičce byla některá pole modifikována



Encapsulating Security Payload (ESP)

* Protokol, který zajišťuje důvěryhodnost dat
* ESP také nabízí svoje vlastní autentifikační schéma, jaké je použito v AH
  + Je ale mnohem náročnější na výpočet
  + Neumí autentifikovat vnější IP adresu
  + AH autentifikuje celý datagram, a to obvykle stačí
* Místo toho, aby ESP mělo jen hlavičku, ESP rozdělí svá pole na tři komponenty:
  + ESP Header – obsahuje dvě pole a přijde ještě před zašifrovanými daty
  + ESP Trailer – umístěno za zašifrovaná data
  + ESP Authentication Data – Když je použitá volitelná authentikační metoda ESP, tato část obsahuje Integrity Check Value (ICV), vypočítána je podobně jako u AH





## IPv6 a QoS (Quality of Service)

IPv4 je založeno na posílání jednoduchých paketů. Se všemi pakety je zacházeno stejně – jsou přeposílány s maximálním úsilím podle FCFS (First Come First Serve) principu. Neexistují žádné možnosti, jak kontrolovat parametry toku, jako např. zpoždění, kolísání signálu nebo alokaci pásma.

Proto byly navrženy dvě hlavní architektury pro poskytování datových toků spolu s prioritami a zárukami kvality:

* *Integrated Services –* založeno a paradigmatu, že šířka pásma a všechny příslušné zdroje jsou rezervovány pro tok dat na end-to-end základu
  + Aplikace nahlásí síti svoje kvalitativní požadavky
  + Síť zkontroluje, zdali jsou požadované zdroje dostupné a rozhodne, jestli bude žádosti vyhověno (tzv. Admission Control Phase)
  + Pokud není možné splnit požadavky, připojení je zamítnuto
  + Pokud je možné požadavky splnit, síť informuje všechny komponenty na cestě k příjemci o nutných rezervacích požadavků (velikost front, jejich priorita atp.)
  + Musí být použit rezervační protkol, např. Resource reSerVation Protocol (RSVP) nebo YESSIR (YEt another Sender Session Internet Reservations)
  + Hlavní nevýhoda: je nutné udržovat stav v uzlech vnitřní sítě (=> problémy se škálovatelností)
* *Differentiated Services –* založeno na označování paketů (přiřazování speciální třídy priority paketům a jejich obsluhování v uzlech vnitřní sítě)
  + Přesná definice požadovaných QoS parametrů není vždy nutná
  + Obvykle dostačuje záruka, že kvalita přenosu se nezhorší, když dojde k přetížení sítě.
  + Differentiated Services – není nutné informovat síť o požadavcích na kvalitu přenosu (=> žádné protokoly pro rezervaci zdrojů)
  + Každý paket je označen značkou priority před tím, než je poslán do sítě
    - Pakety jsou označeny, když vstupují do sítě
    - Značka je přidána do Type of Service pole (IPv4) nebo do Traffic Class (IPv6) pole
    - Uzly vnitřní sítě přečtou značku priority a zpracují paket podle jeho třídy priority
  + Výhody:
    - Jednoduchost
    - Žádné informace o stavu v uzlech vnitřní sítě (=> dobrá škálovatelnost)

Dvě pole IPv6 hlavičky můžou být použita pro QoS:

* **Traffic Class** (= Priorita paketj, PRI) pole
  + Jednobytové pole
  + Zavádí pojem „DiffServ pole“ pro Traffic Class pole
  + DiffServ routery mají známou sadu rutin, které jsou vybrány pomocí šestibitové hodnoty DiffServ CodePoints (DSCP)
    - může být specifikováno až 64 různých kódových hodnot
    - kódová pravidla jsou přiřazována autoritou IANA (Internet Assigned Numbers Authority)
    - Tyto DSCP hodnoty specifikují, jak by měly být pakety přeposílány
  + Poslední 2 bity (Explicit Congestion Notation – ECN hodnota) umožňuje 4 možné kódové hodnoty, které se používají pro Congestion Notification
    - Congestion Experienced bit + ENC-Capable Transport bit
    - Při jejich používání může router signalizovat přetížení ještě před tím, než dojde ke ztrátě paketů
* **Flow label** pole
  + 20-bitové pole
  + Umožňuje klasifikaci paketů náležejících do speciálního *flow*
  + Flow (tok) = posloupnost souvisejících paketů, která je poslána ze zdroje do unicast, anycast nebo multicast destinace
  + IPv6 routery musí obsluhovat všechny pakety náležející do jednoho toku stejně.
    - Když routery obdrží první paket nového toku, můžou zpracovat informaci uvedenou v hlavičkách (IPv6 hlavička, Routing hlavička a Hop-by-Hop extension hlavička) a uložit výsledek do paměti cache
    - Tato informace může být použita ke směrování všech ostatních paketů náležejících do stejného toku
  + Jak využít toto pole efektivně je stále otevřená otázka
  + Klasifikátory toku byly dříve založeny na tzv. 5-tuple: zdrojové a cílové adresy, porty a typy transportních protokolů – méně efektivní, některá z těchto polí mohou být dokonce nedostupná kvůli fragmentaci nebo šifrování
  + IPv6 využívá jen trojici Flow Label a src/dst (destination) address field
  + Všechny toky dat (např. TCP připojení, streamy v aplikacích) musí být označeny, dokonce i když nevyžadují žádná flow-specific zacházení

## IPv6 transition (přechody)

### Portovaní aplikací

Aplikace, které využívají IPv4 síťové sockety, musí být převedeny na IPv6. Programátor musí rozhodnout, zdali program bude pouze IPv4, jestli bude mít oddělené IPv4 a IPv6 nebo bude verze IP protokolu nezávislá (doporučeno ve většině případů).

Problémy, které je třeba vyřešit:

* **Parsování adresy**: IPv4 adresy lze parsovat jednoduše, ale IPv6 adresy potřebují pro parsování podporu od speciální knihovny pro vstup nebo výstup
* **Address memory space**: Starší kód ukládá IPv4 adresy v 32 bitových proměnných typu *unsigned int*, málo strojů má 128 bitový datový typ pro IPv6 adresu.
* **URL a textová reprezentace IP adres:** původní standardy nedovolují užití IPv6 adres jako URL – problémem je dvojtečka v hexa-colon („hexa-dvojtečkové“) notaci, která se v případě IPv4 notace používá pro specifikaci portu
  + Řešení: IPv6 adresy v závorkách: [http://[FE80::219:D1FF:FE06:E908]:8080/](http://[fe80::219:d1ff:fe06:e908]:8080/)
* **Vícenásobné adresy:** V IPv4 měla drtivá většina systému jen jednu adresu na každé rozhraní, ale systémy podporující IPv6 dokáží obsluhovat více IP adres na jedno rozhraní (např. jednu IPv4 adresu, jednu globální IPv6 adresu a jednu link-local IPv6 adresu)

### Interoperabilita a soužití

1. **Dual Stack** – zařízení podporuje IPv4 a IPv6 současně
   1. Umožňuje IPv4 a IPv6 koexistovat současně na jednom zařízení a v jedné síti
   2. Výhody:
      1. Jednoduché využití a flexibilita
      2. Host může komunikovat s IPv4 hosty použitím IPv4 nebo komunikovat s IPv6 hosty za použití IPv6
      3. Jakmile je vše upgradováno na IPv6, IPv4 stack lze jednoduše odebrat
      4. Nabízí největší flexibilitu v rámci práce s aplikacemi, které pracují jen na IPv4
   3. Nevýhody:
      1. Musí zároveň běžet dva stacky protokolů – vyšší nároky na zdroje
      2. Všechny aplikace musí být schopny rozhodnout, zdali host komunikuje pomocí IPv4 nebo IPv6
      3. DNS resolver musí být schopný rozhodnout oba typy adres – Ipv4 i IPv6
      4. Směrovací protokol se musí vypořádat s oběma protokoly
2. **Tunelování** – Ipv6 datagramy jsou zapouzdřeny do dat IPv4 datagramů
   1. To umožňuje přenos IPv4 dat přes existující IPv4 infrastrukturu
   2. Výhody:
      1. Můžeme stále používat IPv4 a aplikace založené na IPv4
      2. Jakmile se vše upgraduje na IPv6, není potřeba provést žádné změny
   3. Nevýhody:
      1. Na routery je kladena dodatečná zátěž
      2. End-pointy tunelů představují jednotlivé body, kde může dojít k selhání
      3. Tunely nabízí možnost bezpečnostních útoků
      4. Problémy s fragmentací
3. **Translators** (NAT-PT) – zařízení překládá IPv6 datagramy na IPv4 datagramy
   1. To umožňuje IPv6 uzlům komunikovat s IPv4 uzly
   2. Výhody:
      1. IPv6 host může přímo komunikovat s IPv4 hostem (a naopak)
      2. Jednoduché dočasné řešení, pokud není možné využít žádnou jinou techniku
   3. Nevýhody:
      1. Nepodporuje pokročilou funkcionalitu IPv6 (jako je např. end-to-end security)
      2. Translators (překladače) jsou ztrátové – Ipv6 má jiné vlastnosti než IPv4
      3. Představuje omezení na návrh topologie
      4. Odpovídá přes stejný NAT router, přes který byly poslány odpovědi
      5. NAT router představuje bod selhání
      6. Všechny aplikace, které mají IP adresu v payloadu s pakety, selžou

# Pokročilé mechanismy směrování

## Distance Vector Routing protokoly

### Routing Information Protokol (RIP)

* Verze:
  + RIPv1 – první směrovací protkol používaný v sítích založených na TCP/IP
  + RIPv2 – přidává několik vylepšení (např. explicitní maskování a autentifikaci směrovací informace)
  + RIPng – rozšíření RIPv2 pro podporu IPv6 adres a sítí
* Metriky – počet hopů (=přenos paketu mezi dvěma sousedními routery)
* Routery posílají informaci periodicky každých 30 sekund
  + Zprávy jsou poslány pomocí UDP protokolu
  + Podporovány jsou spouštěné updaty, když dojde ke změně spojení
  + Timeout je nastaven 180 s
* Vhodné pro malé sítě a stabilní spojení
* Nedoporučováno pro redundantní síť

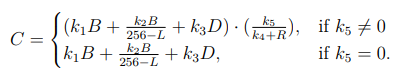
**RIPv1:**

* Broadcast zprávy se skládají z polí:
  + Příkaz – indikuje, zdali je zpráva žádost (router žádá svého souseda o DV (distance vector) informaci) nebo odpověď
  + Verze – verze RIP protokolu
  + Address family identifikátor – 2 pro rodinu adres
  + IP adresa – cílová síť (identifikována podsítí nebo hostem)
  + Metrika – počet hopů do cílové destinace (číslo od 1 do 16, 16 = nekonečno)
* Problémy:
  + Pomalá konvergence a problémy se směrovacími cykly/smyčkami – vynuceno přístupem DV
  + Nekončeno = 16 => RIPv1 nemůže být použit pro sítě s minimálním počtem skoků mezi dvěma routery větším než 15
  + Neexistuje způsob, jak indikovat cokoliv specifického o síti, která je adresována

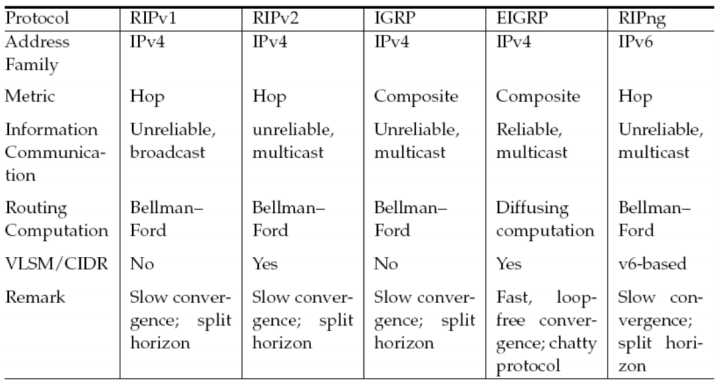
**RIPv2:**

* Multicast zprávy na adrese 224.0.0.9 obsahují nová pole:
  + Route tag – používáno pro odlišení interních cest v rámci RIP směrovací domény od externích cest
  + Subnet mask (maska podsítě) – umožňuje směrování, které je založeno na podsíti místo třídového směrování
  + Next hop – propagovaný router může chtít indikovat další hop, který je různý od sebe sama

### Interior Gateway Routing Protocol (IGRP)

* Vyvinut společností Cisco, aby šlo překonat limit počtu hopů a metriku počtu hopů v RIPv1
* Oproti RIPv1 se liší takto:
  + DV updaty zahrnují pět rozdílných metrik pro každý směr/cestu
  + Běží přímo na IP protokolu (type field je nastaven na hodnotu 9)
  + Umožňuje více cest pro směr za účelem vyvážení zátěže
  + Externí směry můžou být propagovány
* Nepodporuje proměnlivou délku masky podsítě
* Multicast zprávy na 224.0.0.10 obsahují tato pole:
  + Destination – cílová síť, pro kterou je vygenerován Distance Vector (jen 3 B se používají)
  + Delay, Bandwidth, Reliability, Load – Zpoždění, šířka pásma, spolehlivost, zátěž – pole pro výpočty složených metrik
  + Hop count – počet hopů (skoků) – číslo mezi 0 a 255, které indikuje počet hopů do cílové destinace
  + MTU – nejmenší MTU jakéhokoliv spojení na cestě do cíle
* IGRP dává předost použití složené metriky před užitím počtu hopů:
  + To vede k lepším a přesnějším cestám
  + Založeno na čtyřech faktorech: bandwidth (B), delay (D), reliability (R), load (L)
  + Spolu s nezávaznými reálnými koeficienty (k1, …, k5) pro vážení těchto faktorů:
    - Všechny tyto hodnoty jsou nataveny stejně na všech routerech
    - Zpráva protokolu zahrnuje všechny odlišně metriky komponent spíše než složené metriky – ty počítá router.
  + Složené metriky, C („cost of a link“ = „cena linky“), jsou dány následovně
  + 
  + představuje spolehlivost spojení, potom když *k5* = 0, všechny spojení mají stejnou úroveň spolehlivosti
  + Výchozí hodnoty: *k1 = kš = 1* a *k2 = k4 = k5 = 0*, takže *C = B + D*
* IGRP je dále rozšířeno protokolem **Enhanced IGRP** (Rozšířené IGRP) – složená metrika jako u IGRP, poskytuje bez-smyčkové směrování, spolehlivé doručování, umožňuje proměnlivou délku podsíťového maskování

### Porovnání Distance Vector protokolů

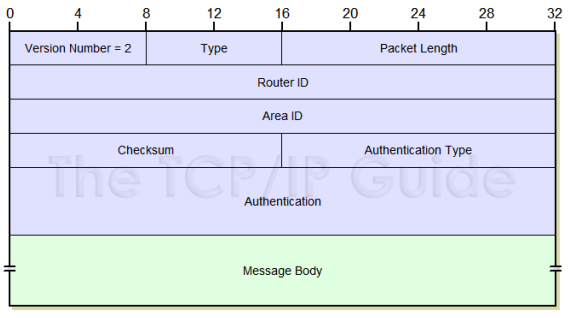


* Split horizon: neposílat informace tomu, od kterého jsme obdrželi informaci

## Link State směrovací protokoly

### Open Shortest Path First (OSPF)

* V současnosti nejpoužívanější Link State protokol
* Sbírá informace o stavu spojení z dostupných routerů a konstruuje topologickou mapu sítě
* Metriky: *cost (=cena)*
  + NO hop-count
  + Číslo (1 – 65535) přiřazené každému síťovému rozhraní routeru
  + Čím je číslo nižší, tím je síťová cesta lepší (bude spíš preferována)
  + Defaultně je každému rozhraní automaticky přiřazena cena, která je vypočítána z propustnosti linky
    - Cena = 100000000/šířka pásma
    - Může být manuálně upraveno
* Autentifikace zprávy – až do OSPFv2, OSPFv3 (který běží na IPv6) již déle nepodporuje autentifikaci uvnitř protokolu (místo toho se spoléhá na zabezpečení protokolu IPv6 (IPsec))
* Směrovací oblasti – další vrstva hierarchie – autonomní systémy mohou být rozděleny na subdomény (směrové oblasti = routing areas) za účelem zjednodušení administrace, optimalizace síťového provozu a využití zdrojů
* Vyvážení zátěže (=Load balancing) – OSPF může využít více odchozích spojení se stejnou (nejnižší) cenou, tzv. Equal-Cost MultiPath (ECMP)
* CIDR/podpora proměnné délky masky podsítě
* OSPF zprávy jsou zapouzdřeny přímo do IP diagramů (protokol číslo 89)
  + OSPF se samo stará o detekci svých chyb a o opravné funkce
  + Pro doručování OSPF zpráv se používá Multicast (224.0.0.5 a 224.0.0.6 pro IPv4, FF02::5 a FF02::6 pro IPv6)



**OSPF Zprávy:**

* Hello paket
* Database Description paket
* Link State Request paket
* Link State Update paket
* Link State Acknowledgement paket

### Intermediate Systém To Intermediate Systém (IS-IS)

* Standardizováno ISO normou jako mechanismus pro komunikaci mezi síťovými zařízeními
* Původně navrženo pro služby síťové vrstvy ISO OSI, které se nazývají CLNS (Connectionless Network Service)
* Později rozšířeno na podporu směrování IP datagramů – nazýváno Integrated (nebo Dual) IS-IS

**Klíčové podobnosti s OSPF:**

* Poskytuje síťovou hierarchii přes dvouúrovňové oblasti
* Používá Hello pakety pro počáteční vytvoření spojení se sousedy a poté pokračuje v udržování těchto spojení
* Podporuje proměnlivou délku masky podsítě
* Udržuje databázi stavů spojení (link state database) a vypočítává nejkratší cestu pomocí Dijkstrova algoritmu

**Klíčové odlišnosti oproti OSPF:**

* Zatímco OSPF pakety jsou zapouzdřeny v IP datagramech, IS-IS pakety jsou zapouzdřeny přímo do rámců linkové vrstvy
* Tím, že IS-IS běží na L2, je vrstva L2 relativně bezpečná před útoky
* IS-IS může směrovat libovolnou síťovou adresu – může se jednoduše přizpůsobit na IPv6 (OSPF potřebuje upgrade na OSPFv3, aby mohlo IPv6 podporovat)
* IS-IS umožňuje deklaraci přetížení – přetížený router by neměl být zahrnut do výpočtu cesty v síti
* Metriky spoje OSPF jsou v rozmezí od 1 do 65535, zatím co metriky IS-IS jsou v rozmezí od 0 do 63 (užší metrika) a můžou být dále rozšířeny až do rozmezí od 0 do 16777215 (široká metrika)
* OSPF poskytuje bohatší sadu rozšíření a přidaných vylepšení
* IS-IS je méně „upovídané“ a může být škálováno za účelem podpory rozsáhlejších sítí

## Path Vector Routing Protocol

### Border Gateway Protocol (BGP)

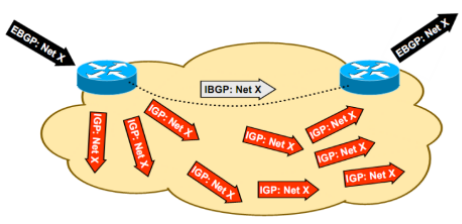
* Aktuální je verze 4 (BGP-4)
* Navrženo za účelem růstu Internetu a složitosti (podporuje redundantní topologie, umí vyřešit smyčky/cykly, …)
* Používá se pro posílání informací o umístění sítě z jednoho autonomního systému do jiného autonomního systému
  + Výměna je provedena nastavením komunikace mezi sousedícími autonomními systémy
  + Komunikační kanál je nastaven na vrcholu TCP protokolu (plně spolehlivé)
* Umožňuje definici směrovacích pravidel
* Využívá hop count metriky
* Využívá CIDR pro agregaci cest v síti
* BGP peery posílají *advertisementy* navzájem mezi sebou
  + Advertisement = cílová síťová adresa (využití CIDR notace) + atributy cesty (AS na cestě, next-hop router atd.)
  + Odesláno přes spolehlivé point-to-point komunikační kanály
* Jakmile je cesta oznámena AS, přichází na řadu *routovací (směrovací) pravidla*
  + Routovací pravidla definují, které AS mají povoleno přeposílat data přes konkrétní AS, do kterých AS mohou být data přeposílána atd.
  + Neexistují žádné standardy
  + Pokud nejsou routovací pravidla definována, je vybrána nejkratší cesta

**Typy BGP zpráv:**

* OPEN – iniciuje BPG session mezi párem BGP routerů. Routery se představí a oznámí své schopnosti a autentifikační informace.
* UPDATE – oznamuje routovací informaci z jednoho BGP routeru do jiného („push model“). Stáhne předchozí oznámené oznámení – oznamovací informace je validní do té doby, dokud není explicitně stažena.
* KEEPALIVE – posíláno, pokud není žádný jiný síťový provoz. Umožňuje BGP routerům rozlišit mezi selháním spojení a BGP peerem, který nemá co posílat.
* NOTIFICATION – používáno pro ukončení sezení nebo k reportování chyby (např. odmítnutí OPEN zprávy nebo nahlášení problému s UPDATE zprávou)
* ROUTE-REFRESH – žádot o znovu-oznámení všech síťových cest v routovací tabulce routeru za pomocí UPDATE zprávy. Tento typ zprávy není v původní BGP-4, přidán až později normou RFC.

**Internal (vnitřní) BGP (IBGP)**

* Umožňuje, aby externí destinace (AS) byly dosažitelné ze všech routerů v rámci AS.
* Poskytuje informace o přilehlých AS interním routerům v konkrétním AS
* Všechny IBGP peery v rámci stejné AS jsou plně propojené
* Peer oznámí cestu, kterou obdržel přes eBGP (external BGP), IBGP peerům
* Ale: IBGP peery neoznamují cesty přijaté přes jiné IBGP peery
* Naučené cesty jsou distribuovány dále přes IGP



## Architektura routerů

### Funkce routerů

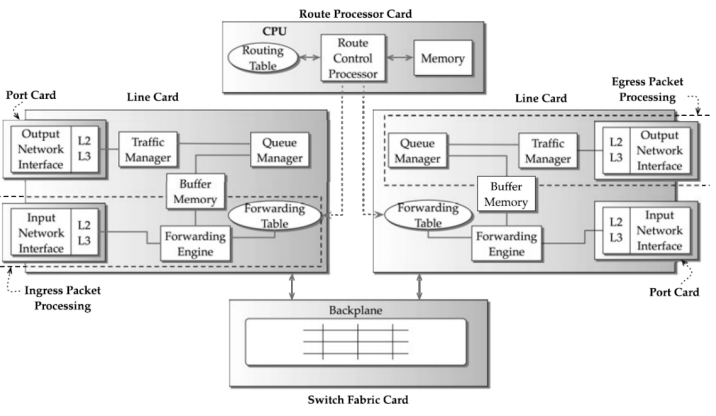
Router musí umět provádět dva základní úkony: routing (směrování) a přeposílání paketů (packet forwarding)

1. *Routing proces* sestavuje náhled topologie a vypočítává nejlepší cestu v síti
   1. Na základě informací vyměněných mezi sousedními routery s využitím routovacích protokolů
   2. Nejlepší cesty jsou uloženy v datových strukturách nazvaných *forwarding table*
2. *Packet forwarding proces* přesunuje pakety ze vstupního rozhraní („ingress“) na příslušné výstupní rozhraní („egress“)
   1. Na základě informací obsažených ve forwarding table
   2. Výkon forwarding procesu ovlivňuje celkový výkon routeru

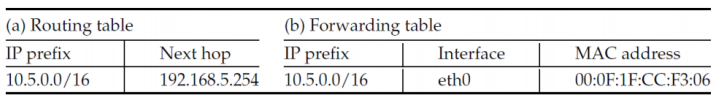
**Dodatečné funkce:**

* Routovací protokoly: Routery musí implementovat různé routovací protokoly (např. OSPF, BGP a RIP), aby mohly udržovat vztahy mezi jednotlivými peery tím, že odesílají a přijímají updaty síťových cest od příslušných routerů
* Konfigurace systému: Routery musí implementovat různé funkce, které umožní operátorům konfigurovat různé administrativní úkoly (nastavování rozhraní, udržování routovacího protokolu při životě, pravidla pro klasifikaci paketu atd.)
* IP header validace: Každy IP paket přicházející na router musí být validován (např. číslo verze protokolu musí být správně, délka hlavičky paketu musí být správně, musí sedět kontrolní součet atd.)
* Packet Lifetime Control: Snížení TTL pole kvůli předcházení uváznutí paketů ve routovací smyčce (pokud je TTL menší nebo rovno nule, je paket zahozen a je vygenerována a poslána (původnímu odesílateli) ICMP zpráva
* Přepočítávání kontrolního součtu: Jakmile je hodnota TTL modifikována, musí být aktualizována i hodnota kontrolního součtu v hlavičce
* Route Lookup: Cílová adresa paketu se používá pro vyhledávánní ve forwarding table za účelem určení výstupního portu
* Fragmentace: Router musí rozdělit pakety do několika fragmentů, pokud je MTU odchozího spojení menší než velikost paketu, který má být přenášen
* Obsluha IP Options: Paket může indikovat, že vyžaduje na routeru vyžaduje zvláštní potřeby
* Klasifikace paketů: pro rozlišování paketů router nemusí přezkoumat jen cílovou IP adresu, ale také ostatní pole jako je např. zdrojová adresa, cílový port, zdrojový port atd.
* Překlad paketů: Router, který se chová jako brána do NAT sítě, musí podporovat překlad síťových adres
* Prioritizace síťového provozu: Router může potřebovat záruku určité kvality služby, aby splnil dohodnutou úroveň poskytovaných služeb,

### Router Elements



* Síťová rozhraní: Síťové rozhraní obsahuje mnoho portů, které poskytují připojení ke spojením fyzické sítě. Port je specifický pro konkrétní typ síťového fyzického média (Ethernet, Sonet atd.)
* Forwarding Engines: Rozhodnutí, na které síťové rozhraní by měl být příchozí paket přeposlán – s ohledem na forwarding table = Address/Route Lookup
* Queue Manager: poskytuje vyrovnávací paměť pro dočasné uložení paketů, když dojde k přeskupení odchozího spojení z routeru. Pokud dojde k přetečení této dočasní paměti kvůli zahlcení, manažer fronty zahodí vybrané pakety.
* Traffic manager: Je zodpovědný za prioritizaci a regulování odchozího síťového provozu v závislosti na požadované úrovní služeb
* Backplane: Poskytuje připojení k síťovým rozhraním. Pakety z příchozího síťového rozhraní můžou být přesunuty do odchozího síťového rozhraní.
* Route Control Processor:
  + Zodpovědný za implementaci a vykonávání routovacích protokolů. Udržuje routovací tabulku, která je aktualizována vždy, když dojde ke změně síťové cesty (route) na základu obsahu routovací tabulky je vypočítána a aktualizována forwarding table.
  + Spouští software, který konfiguruje a kontroluje běh routeru
  + Provádí komplexní packet-by-packet operace (zpracování chyb během práce s pakety, odesílání ICMP zpráv k odesílateli, když např. nemůže být nalezena cílová destinace paketů ve forwarding table atd.)



### IP Address Lookup Algoritmy

V případě použití **classful** adresovacího schématu je přeposílání paketů jednoduché. Routery potřebují prozkoumat jen *address class* a *network part* cílové adresy. Tím pádem si forwarding table potřebuje ukládat jen jednoduchý záznam pro směrování paketů určených pro všechny hosty připojené k dané síti.

Address lookup s CIDR je složitější, protože:

* Cílová IP adresa nenese přímo netmask informaci
* Prefixy ve forwarding table, se kterými musí cílová adresa být shodná, mohou být libovolné délky => **Longest Prefix Matching** (shoda nejdelšího prefixu)

Požadavky pro Longest Prefix Matching

* **Lookup Speed:** Měření internetového provozu ukazují, že zhruba 50 % všech paketů, které dorazí do routeru, jesou 40 B TCP-ack pakety. Tím pádem se prefix lookup musí provést v čase, který je tak dlouhý, jako trvá odeslání paketu nejmenší možné velikosti (40 B) – tzv. *wire-speed forwarding*
  + Pro linku 1Gbps by prefix lookup nemělo být delší, než
* **Memory Usage** (Využití paměti): Algoritmus by měl efektivně využít rychlou ale malou cache paměť
* **Škálovatelnost:** Algoritmy by měly být škálovatelné jak v rychlosti, tak co se týče paměti, a to podle toho, jak se zvětšují forwarding tables.
* **Updatability** (schopnost aktualizace): Změna cesty v síti (route) se stává celkem často (až stovky prefixů za vteřinu) => aktualizace forwarding table by neměla trvat déle než několik ms.

Algoritmy pro Longest Prefix Matching:

* Lineární vyhledávání prefixů: *O(n)*, kde *n* je počet prefixů ve forwarding table
* Trie-based Algoritmus (trie = vyhledávání)
* Hledání pomocí Length Algorithms
* Hledání pomocí Value Algorithms
* Hardware Algorithms (např. RAM-Based Lookup)

### IP Packet Filtering a klasifikace

1. Poskytuje různé záruky služeb pro různé typy síťového provozu. ISP může udržovat různé cesty pro stejné zdrojové a cílové adresy.
2. Flexibilní účtování dat (různé typy síťového provozu můžou být zpoplatněny různě)
3. Ochrana před škodlivými útoky – škodlivé pakety mohou být identifikovány a zahozeny už na počátku

* Pravidla/politika = kritéria pro klasifikaci paketů
  + Použitím polí z hlavičky paketů
  + Přeposílací mechanismus potřebuje vyšetřit i jiné pole paketů, než je cílová adresa, aby identifikoval kontext paketů a provedl tak potřebné akce za účelem splnění požadavků uživatele
* Sada takových pravidel/politik – databáze pravidel/politiky = klasifikátor toku = klasifikátor
* Každé pravidlo specifikuje:
  + Tok, do kterého paket náleží (podle vyjádřených podmínek) – přesná shoda, shoda prefixu, shoda rozsahu, shoda podle regulárního výrazu atd.
  + Akce, která má být spuštěna nad pakety náležící do toku – povolit, zakázat, zašifrovat atd.
* Paket může odpovídat více než jednomu pravidlu v klasifikátoru
  + Cena je spojena s každým pravidlem pro rozhodnutí jednoznačné shody
  + Cílem je najít pravidlo s nejnižší cenou, které je shodné s hlavičkou paketu
  + Když jsou pravidla seřazena podle jejich ceny, je cílem najít pravidlo se shodou co nejdříve

Algoritmy:

* Naivní: ukládání pravidel do spojovaného seznamu za účelem zvýšení ceny (úložiště je efektivní, ale vyhledávací doba není)
* Dvourozměrné řešení – Hierarchické stromy
* D-rozměrné řešení
* Metoda rozděl a panuj (Lucent Bit Vector, Aggregate Bit Vector,, Cross-Producting, Recursive Flow Classification)
* Hardwarově založené řešení

## Traffic Engineering v IP sítích

(Interní) Routovací protokoly používané v IP sítích jsou založeny na principu Shortest Path First (SPF) směrování. V nepoužívané síti je tento princip ideální – datagramy jsou doručeny s co nejmenším využitím síťových zdrojů. Nicméně jakmile se zvýší síťový provoz, routery na nejkratší cestě mohou být přehlceny zatím co delší cesty zůstanou nevyužity.

Equal-Cost MultiPath (ECMP) je použitelné, ale neřeší daný problém.

*Traffic Engineering* je celé o objevování jaké další cesty a spojení jsou k dispozici v síti, jaké je aktuální využití síťového provozu, směrování síťového provozu na jiné než nejkratší cesty, takže je dosaženo optimálního využití zdrojů.

* Dosaženo kombinací těchto prvků: rozšíření existujících IGP protokolů + nástroje pro monitorování síťového provozu + techniky síťového směrování
* Výskyt mimo aktuální síť

Kroky:

1. Měření síťového provozu jsou sbírána k odhadnutí síťové matice
2. Topologie a konfigurace je získána ze sítě
3. Proces určování váhy spojení rozhoduje váhu spojení na základě výstupů z bodu 1. a 2.
   1. Vypočítané váhy spojení pro každé jedno spojení jsou poslány do sítě, tzn. každý router přijme metriky pro své odchozí spojení
   2. Jakmile jsou váhy poslány, za využití standartního OSPF/IS-IS procesu „zaplavování“ jsou metriky rozšířeny přes link-state zpráv

Jak často by měl TE systém aktualizovat váhy spojení?

* Záleží na poskytovateli sítě/administrátorovi
* Obvykle jednou za den až týden, aby se předešlo krátkodobým kolísáním síťového provozu

### Krok 1: Zjišťování využití sítě

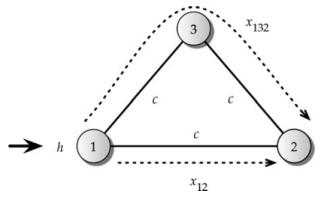
* Je potřeba mít široký pohled na síť v případě využití síťových zdrojů.
* Existuje několik metod ke sbírání informací o využití sítě:
  + **SNMP**: Aplikace posílá signál na každý router a informaci, která se vrátí zpět, konvertuje na náhled využití sítě. Nerozhoduje, který tok dat musí být rozšířen znovu, aby došlo ke zmírnění přetížení (je získána je absolutní míra zátěže sítě)
  + **NetFlow**: Nástroj od Cisco, který sbírá informace na klíčových místech v rámci sítě. Zahrnuty jsou agregační body (NetFlow collectors) pro upevnění informací z podmnožiny celé sítě.
  + **sFlow, ntop** atd.
* Sítový provoz je nestabilní a (obvykle časově závislý)

### Krok 2: Zjišťování topologie sítě

* Aplikace, která dělá TE rozhodnutí, musí mít jasný pohled na topologii sítě a schopnosti spojení v rámci sítě.
* Malé statické sítě – manuální konfigurace je dostatečná
* Větší a dynamické sítě – musí být použit automatický systém
  + IGP protokol je rozšířen, aby mohly být rozšířeny dodatečné informace o spojeních
  + OSPF a IS-IS byly rozšířeny, aby poskytovaly (pro každé spojení): metriky traffic engineering, maximální šířku pásma, maximální rezervovatelnou šířku pásma, nerezervovanou šířku pásma atd.

### Krok 3: Výpočet váhy spojení

Komodita ~ požadavek na kapacitu linky. **Single-commodity** – jeden jediný pár uzlů v síti má pozitivní požadavek na objem dat.

* Uvažujme následující síť:
  + *c –* kapacita každého spojení (zde stejné pro všechna spojení)
  + *h –* požadovaný objem pro pár uzlů 1 : 2
  + *x12, x132 –* množství požadovaného objemu dat, které musí být směrováno
* Následující omezení musí být splněna:
  + Požadovaný objem dat *h* musí být přenesen přes tyto dvě cesty: *x12 + x132 = h*
  + Cesta nemusí nést žádný negativní požadavek: *x12 >= 0, x132 >= 0*
  + Žádný tok dat na cestě nemůže překročit kapacitu na kterémkoliv spojení, které cestu využívá: *x12 <= c, x132 <= c*

Předpokládejme cíl minimalizace ceny směrování síťového toku:

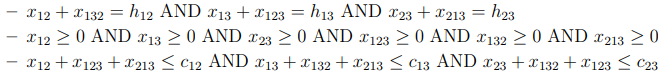
* *Cost per unit of flow* (cena za jednotku toku – cena zaplacená za přenesená data) – na každé cestě: *p12 >= 0, p132 >= 0*
* Celková cena (objektivní funkce): *F = p12x12 + p132x132*
  + Přidáme tři omezení, která musí být splněna a dostaneme problém lineárního programování – systém rovnic, jehož řešení vyhovuje daným podmínkám
  + Výsledný systém řeší zadaný cíl – minimalizaci ceny směrování pro zadanou topologii, kdy požadavky na síťový provoz *h* jsou dány

Nyní předpokládejme cíl minimalizace maximálního využití linky:

* Využití linky 1 – 2:
* Využití linky 1 – 3:
* Maximální využití přes všechny linky:
  + Spolu s třemi omezeními dostaneme soustavu rovnic, která řeší vyvážení zátěže na cestách 1 – 2 a 1 – 3 – 2, kde požadavek na síťový provoz *h* je daný.

**Multicommodity –** všechny tři páry požadavků můžou mít pozitivní požadavek na objem dat *h12, h13, h23*

* Pro každý požadovaný pár může být objem požadavku naplněn využitím dvou cest: např. 1 – 3 a 1 – 2 – 3 pro pár požadavku 1 : 3
* Omezení:



* Celková cena: 

Poznámka: V IP sítích založených na OSPF ne IS-IS jsou nejkratší cesty vypočítány na základě váhy linků (spojení) – tento výpočet Nezahrnuje objem síťového přenosu nebo (obvykle) kapacitu sítě. Naproti tomu, předchozí příklad Nebere v potaz váhu linků (spojení).

## Multiprotocol Label Switching (MPLS)

MPLS je nový mechanismus pro forwarding (přeposílání), původně prezentovaný jako způsob vylepšení rychlosti forwardingu centrálních IP směrovačů

V MPLS síti jsou pakety přeposílány na základě tzv. *labels* (štítků):

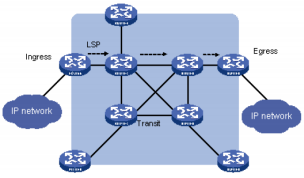
* Štítek je přidán na začátek paketu, jakmile paket vstoupí do MPLS sítě (tzn. přidán jako další hlavička, takže router ví, jak se na základě takového štítku chovat)
* Interní MPLS routery neprohlíží IP adresy paketů – krátký pohled na štítek fixní délky je mnohem rychlejší, než hledání shody nejdelšího prefixu (předpony) na každém routeru.
* Štítky obvykle odpovídají sítím IP destinace, ale mohou také odpovídat jiným parametrům, jako např. QoS nebo zdrojová adresa
* Vyžaduje nové protokoly (nebo rozšíření stávajících), aby se informace o štítcích mohly rozšiřovat.
* Štítky se mohou během života v síti měnit
* MPLS toky dat jsou connection-oriented (pro každý tok dat musí být nastaveno nějaké spojení) a pakety jsou směrovány po předem nastavených Label Switched Paths (LSPs)
  + MPLS spojení je jednosměrné
  + Dvousměrná komunikace vyžaduje založení páru LSP (cesty pro přeposílání a vracení se mohou lišit)
* MPLS umožňuje nové forwarding paradigmata, která nebyla dostupná pro IP směrování, např. schopnost síťových operátorů diktovat cesty, které bude síťový provoz využívat v jejich síti (data s nízkou prioritou můžou cestovat po delší cestě, aby zůstala nejkratší cesta volná pro data s vysokou prioritou)
* MPLS se stalo zásadní a standartní technologií pro rozsáhlé IP sítě

**Základní funkcionalita**

1. Pakety, které vstupují do sítě, jsou analyzovány a klasifikovány podle FEC (Forward Equivalence Classes) a základě cílové IP adresy, typu služby atd.
2. Pro všechny FEC třídy jsou vytvořeny štítky
3. Jsou vytvořeny LSP
4. Štítky jsou rozeslány
5. Tabulky s informacemi o přeposílání, známe jako LFIB (Label Forwarding Information Base), jsou nastaveny. Tabulky mapují pár (*incoming\_label*, *incoming\_interfac*) na pár (*outgoing\_label, outgoing\_interface*). Nový štítek nyní může být přidán na paket, který opouští router.
6. MPLS hlavička (tzv. shim header) je vytvořena. Pakety budou přeposlány a základě jejich štítků, které jsou zapsány v jejich hlavičce.

**Komponenty sítě**

* Okrajové Label-Switched Routers (Edge-LSRs) = hraniční routery
  + Ingress-LSR: (vstupní LSR) analyzují informace v IP hlavičce paketu. Na základě těchto analyzovaných informací je paket přiřazen konkrétnímu FEC. V závislosti na přiřazeném FEC je do MPLS hlavičky vložen správný štítek.
  + Egress-LSR: (výstupní LSR) odstraní MPLS hlavičku a pošle původní IP paket do výstupního spojení. Dále také sníží TTL pole paketu.
* Centrální Label-Switched Routers (Core-LSRs): zajišťují přeposílání paketů na základě přiřazeného štítku. IP hlavička není Core-LSR routerem ani upravena, ani analyzována. Analyzovány a modifikovány jsou v případě nutnosti jen MPLS štítky.



**MPLS Shim header**

* Hodnota štítku
* Třída síťového provozu
* Zásobník – nastaven na 1 pro poslední záznam v zásobníku štítku a na hodnotu 0 pro všechny ostatní záznamy. Přijímající router zkoumá jen nejhornější štítek.
* TTL – Zakóduje Time To Live hodnotu pro daný MPLS cloud



* Obvykle je paketu přiřazen jen jediný MPLS štítek
* VPN sítě MPLS vytvoří 2 štítky: horní štítek ukazující na výstupní router a druhý štítek, který identifikuje VPN.
* MPLS Traffic Engineering vytvoří 2 štítky: horní štítek ukazující na endpoint (cílový bod) traffic engineering tunelu a druhý štítek ukazující na cílovou destinaci
* MPLS TE kombinováno s MPLS VPN vytvoří 3 a více štítků

### MPLS distribuce štítků

Před tím, než můžeme použít LSP, LFIB musejí být obsazeny v každém LSR na síťové cestě – je třeba použít protokol pro distribuci štítků

Lze použít několik protokolů: rozšíření BGP (Border Gateway Protocol), RSVP-TE (RSVP-Traffic Engineering) (upravená verze RSVP), LDP (Label Distribution Protocol), TDP (Tag Distribution Protocol, Cisco), LDP/CR (Label Distribution Protocol/Constrained Routing), …

**Přístupy k Label Distribution**

1. *Downstream-on-demand, ordered contol:*
   * Distribuce probíhá ve směru toku
   * MPLS zařízení nesignalizuje FEC-to-label vazbu, dokud o to není požádáno upstream (vstupním) zařízením
   * LSR nerozesílá štítek pro FEC, dokud se nejedná o výstupní LSR pro FEC nebo dokud LSR neobdrželo štítek pro FEC od svého downstream (výstupního) peeru
   * Stejný štítek lze použít pouze mezi přilehlými peery
2. *Dowstream-unsolicited:*
   * Směrovač rozesílá štítek všem ve své síti (blízkým uzlům)
   * MPLS zařízení nečekají na žádost z upstream zařízení před signalizací FEC-to-label vazeb
   * Jakmile LSR zná cestu, pošle vazbu na tuto cestu na LSR všem svým peerům, jak upstream, tak downstream

**Label Distribution Protocol (LDP)**

* Definováno pomocí IETF pro distribuci štítků v MPLS prostředí
* Spoléhá na routing informace poskytované pomocí IGP za účelem přeposílání štítků paketů
* Používá TCP nebo UDP
* Může operovat v režimech Downstream-on-demand a i Downstream-unsolicited
* Hlavní aktivity:
  + Objevení „přilehlých“ LSR schopných LDP (Discovery message)
  + Zřízení kontrolní konverzace mezi přilehlými LSR a vyjednávání možností (Adjecency message)
  + Ohlašování/odebírání štítků (Label Advertisement Message)
  + Chybové hlášení (Notification message)

### Traffic Engineering v MPLS

* MPLS může zřídit LSP, které sleduje jinou cestu, než tu „preferovanou“ směrovacím protokolem a přeposílacím algoritmem.
* Zdroje uvnitř sítě mohou být dynamicky rezervovány, jakmile jsou jednotlivá LSP zřízena a mohou být i dynamicky aktualizovány, když se změní potřeby LSP
* Může být zřízeno více LSP mezi párem zdrojových a cílových koncových bodů (síťový provoz může být upraven na paralelní LSP)
* Můžeme definovat procedury obnovení pro přenos síťového provozu do alternativních LSP v případě, že dojde k selhání.

### Generalized MPLS (GMPLS)

GMPLS přidává funkcionality do MPLS, která je nad rámec pouhého přepínání paketů:

* Schopnost přepínání paketů – např. GMPLS zahrnuje MPLS
* Schopnost Time-Division Multiplexing – pro přepínání časových okruhů
* Schopnost Lambda-Switch – pro přepínání vlnové délky na optických propojeních
* Schopnost Fiber-Switch – pro přepínání úrovní optických vláken

### Grid-enabled MPLS (G2MPLS)

G2MPLS vylepšuje GMPLS tím, že poskytuje jednokrokovou rezervaci zdrojů, kooperovanou alokaci a údržbu jak síťových zdrojů, tak grid zdrojů. Moc se nepoužívá, protože není škálovatelný.

## QoS-Based Routing (Směrování založené na Quality-of-Service)

QoS-Based Routing je:

* směrovací mechanismus, pod kterým jsou cesty pro tok dat rozhodovány na základě alespoň nějaké znalosti o dostupnosti zdrojů v síti, stejně jako QoS požadavků na toky,
* nebo dynamický směrovací protokol, který zahrnuje kritéria pro rozhodnutí cesty jako je např. šířká pásma, spojení a využití end-to-end cesty, využití zdrojů na jednotlivých uzlech, zpoždění, odezva atd.

**Cíle:**

* Dynamicky vyhledat cestu ze zdroje k cíli zároveň, což může splnit požadavky koncového uživatele na šířku přenosového písma nebo end-to-end zpoždění
* Optimalizace využití síťových zdrojů za účelem maximalizace propustnosti v síti
* Minimální degradace výkony sítě v případě zahlcení.

**Problémy:**

* Výpočet síťové cesty. Počet hopů není dostačující, musí být zvolena vhodná metrika (dostupná šířka pásma, zpoždění, jitter)
* Rezervace zdrojů. Jakmile je vybrána použitelná cesta, musí být rezervovány příslušné zdroje (šířka pásma, velikost bufferu v routerech atd.) pro síťový provoz, které tím pádem nejsou dostupné ostatním.
* Propagace znalostí o síti. Oproti metodě Best-effort routingu je třeba sdílet více informací. Metriky používané QoS-based směrováním se mohou měnit velmi rychle. Nicméně pokud dojde ke sdílení informací pokaždé, když se změní hodnoty metrik, může se stát, že routery a síťová spojení budou velmi zatíženy. Obvyklou cestou, jak tento problém řešit, je nastavení vhodné meze pro rozlišení signifikantních změn od malých změn.
* Předpokládá se, že QoS-based směrování je škálovatelné. Za účelem udržení složitosti výpočtu síťové cesty a množství informací, které je třeba sdílet a udržovat pod kontrolou, se používá hierarchická agregace.
* Administrativní kontrola. Různé toky dat v síti by měly mít různou prioritu. Zdroje by měly být alokovány spravedlivě podle všech tříd priorit (abychom se vyhnuli efektu „starvation“ – vyhladovění v případě nízkých priorit).
* Intergace QoS-based směrování a Best-Effort směrování. Některé toky dat v síti nepotřebují být upřednostňovány a ani se nestarají o QoS. QoS-based směrování musí být schopné podporovat best-effort směrování

### Směrovací algoritmy

* **Source-based:** Každý router má globální stavovou informaci o síti, na základě čehož je vybrána lokální cesta. Jakmile je cesta vybrána, zdrojový router uvědomí ostatní routery na této cestě o tom, jak přeposílat síťový tok dat.
  + Výhoda – jednoduchost
  + Nevýhoda – Každý router musí získat kompletní informace o stavu sítě (náročné na udržování). Současně dochází k výpočetnímu přetížení na zdrojových routerech => problémy se škálovatelností (nevhodné pro rozsáhlé sítě)
* **Hop-by-hop**: Každý router zná pouze následující hop směrem k cíli.
  + Výhoda: Více kompatibilní s existujícími směrovacími protokoly, výpočet směrování je rozdělen na všechny routery na síťové cestě
  + Nevýhoda: Může nastat problém routing smyčky (Když nejsou informace v routerech konzistentní). Je také problém se škálovatelností.
* **Hierarchical:** Struktura směrování obsahuje více úrovní. Spodní úroveň se skládá z aktuálních routerů. Tyto routery jsou organizovány do logických skupin, které zase tvoří vyšší úrovně. Každý uzel obsahuje detailní informace o své skupině a také sjednocené informace o ostatních skupinách.
  + Výhodou je škálovatelnost – vhodné pro velké sítě
  + Nevýhoda: Agregace snižuje přesnost informaci o stavu směrování

### Směrovací protokoly (FYI)

**Private Network-Network Interface (PNNI)**

* Hierarchický a dynamický směrovací protokol pro ATM sítě (telekomunikace). Je založen na link-state algoritmu.
* Informace o topologii je rozšířena v síti tzv. „záplavou“. Síťové zdroje jsou definovány atributy a metrikami, které jsou seskupeny podle podporované třídy síťového provozu. Treshold algoritmy (prahové algoritmy) se používají pro rozhodování, jestli je změna v metrikách dostatečně zásadní, aby bylo třeba vyžadovat propagaci aktualizovaných informací.

**QoS směrovací rozšíření protokolu QSPF (QOSPF)**

* Hierarchický směrovací protokol, který je založen na link-state algoritmu.
* Funguje, když jsou potřeba oba typy routování – jak QoS-based, tak best-effort routování. Metriky šířka pásma linky a zpoždění propagace jsou přidány oproti LSA.
* Algoritmus pro každou možnou destinaci předpočítává nejširší a nejkratší cestu (minimální počet hopu spolu s maximální šířkou pásma)
  + Nejširší verze cesty Bellman-Fordova algoritmu pro před-počítání
  + Nejširší a nejkratší verze cesty Dijsktrova algoritmu pro počítání na vyžádání

# Tradiční protokoly TCP

## Tradiční TCP

Vlastnosti protokolů pro spolehlivý přenos dat:

1. **Zajistit spolehlivost přenosu:** Znovuodesílání ztracených paketů, FEC by mělo být užitečně zaměstnáno
2. **Ochrana před zahlcením:** sítě a příjemce

Vyhodnocování chování (porovnání různých protokolů rodiny TCP):

1. **Agresivita** – schopnost využití dostupné šířky pásma
2. **Schopnost reakce** – Schopnost zotavení se ze ztráty paketů. Narůstá spolu s narůstající šířkou pásma. Základní reakcí na ztrátu paketů je jejich znovuodeslání
3. **Spravedlivost** – získání spravedlivé porce síťové propustnosti v případech, kdy více streamů/přispěvatelů využívá síť. Uvažujeme streamy s rozdílným RTT a streamy s rozdílným MTU. Rychlost přibližování k rovnovážnému bodu je důležitá!

**Kontrola toku:** explicitní zpětná vazba od příjemce(ů) s využitím operace *rwnd*. Je deterministická. Aby rychlý odesílatel nezahltil pomalého příjemce.

**Kontrola zahlcení:** Přibližný odesílatelův odhad dostupné propustnosti (využitím operace *cwnd*). Abychom nezahltili síť mezi odesílatelem a příjemcem. Finální použité okno – *ownd*.



Tradiční TCP – Tahoe, Reno, Vegas mají kontrolu zahlcení založenou na **AIMD** – Additive Increase Multiplicative Decrease.

**Tahoe:**

* Pro RTT beze ztrát (vyšší než *sstresh*): *cwnd = cwnd + 1*
* Pro každou ztrátu: *sstresh = 0,5; cwnd = 1;* znovuodeslání celého *ownd*

**Reno:**

* Přidává **rychlé znovuodesílání**:
  + (TCP příjemce odešle okamžitý duplikátní ACK (Acknowledgment příznak), když dorazí out-of-order segment – segment mimo pořadí). Všechny segmenty, které následují za zahozeným segmentem, spustí duplikování ACK.
  + Ztráta je indikována třemi duplikátními příznaky ACK (čtyři postupně jdoucí identické příznaky ACK bez toho, aniž by mezi nimi byl paket). Jakmile příznaky dorazí, TCP provede rychlé znovuodeslání tím, že pošle jeden samotný segment bez čekání na časovač hlídající dobu znovuodesílání.
* Přidává **rychlé zotavení** – žádný pomalý start, *sstresh = cwnd =* 0,5\* *cwnd.*

**Vegas:** Pokouší se předpovídat zahlcení sítě tím, že ji monitoruje – měří dostupnou síťovou šířku pásma pomocí tzv. *inter-packet* *spacing/dispersion* (spacing = řádkování, vzdálenost; dispersion = rozptyl)

* Když je síť zahlcena, zvýší se RTT
* RTT je monitorováno během přenosu
* Když je detekováno zvýšení RTT, velikost okna pro zahlcení je lineárně redukována

## Vylepšení tradičního TCP

1. Multi-stream TCP
   1. Předpokládá více TCP streamů, které přenášejí jeden datový tok
   2. Ve skutečnosti vylepšuje výkon a chování TCP jen v případě, že dojde ke ztrátě izolovaných paketů (ztráta více paketů obvykle ovlivní více TCP streamů)
   3. Masivně používáno – jednoduchá implementace (GridTCP, Internet Backplane Protocol)
   4. **Nevýhody:** komplikovanější než tradiční TCP, spuštění je zrychleno pouze lineárně, vede k souběžnému přetížení front a cache pamětí v routerech.
2. Vylepšení implementace TCP
   1. Je dostupná **spolupráce s HW**
   2. **Zero copy** – přistupování k síti obvykle vede k několika kopiím dat v paměti. Abychom se tomu vyhnuli, umožňujeme tzv. page flipping – „Převracení stránek“. Stránky v paměti po odeslání náleží systému a po přijetí náleží uživatelskému programu.
3. **Web100** je software, který implementuje nástroje balíku Linux TCP/IP. Je rozdělen do dvou částí: 1. rozšíření kernelu (jádra), které přidává nástroje. 2. Soubor „uživatelských“ knihoven a nástrojů pro přístup k nástrojům jádra (příkazový řádek, GUI). Umožňuje monitorování (rozšířené statistiky), vylepšování nástrojů, podpora pro automatické ladění.

## Konzervativní rozšíření TCP

1. GridDT
   1. Kolekce ad-hoc modifikací
   2. Oprava *sstresh* (rychlejší slowstart)
   3. Modifikace AIMD pro kontrolu zahlcení: *cwnd = cwnd* + *a …* pro RTT bez ztráty paketu, *cwnd = b \* cwnd…* pro ztrátu paketu.
   4. Je třeba modifikovat jen stranu odesílatele
2. Scalable TCP (škálovatelné TCP)
   1. Kontrola zahlcení už není AIMD, ale **MIMD** (Multiplicative IMD)
      1. *cwnd = cwnd* + 0,01 \* *cwnd* … na RTT beze ztrát
      2. *cwnd = cwnd* + 0,01 … na jeden příznak ACK
      3. *cwnd =* 0,875 \* *cwnd* … na jednu ztrátu paketu
   2. Pro menší šířky okna a/nebo větší ztrátovost v síti – Scalabel-TCP se přepne do AIMD režimu
   3. Spravedlnost – Scalable kontrola zapnuta, pokud je > 30 MB/s, je třeba 2x více kroků
   4. Časy pro zotavení ze ztráty paketů jsou poměrné jen k připojení RTT
3. High-Speed TCP (HSTCP)
   1. Kontrola zahlcení AIMD/MIMD:
      1. *cwnd = cwnd* + (*cwnd*) na RTT beze ztrát
      2. 
      3. 
   2. Emuluje chování tradičního TCP pro malé velikosti okna a/nebo větší míry ztráty paktů v síti.
   3. Je možná parametrizace ekvivalentní k Scalable-TCP: *Linear HSTCP*
   4. Ani Scalable TCP ani HSTCP si neumí poradit se slow-start fází
4. H-TCP
   1. Vytvořeno v Hamilton Institutu v Irsku
   2. Jednoduchá změna v *cwnd* increase funkci
   3. Využívá se AIMD mechanismus
   4. Zvyšuje agresivitu s tím, jak se prodlužuje čas od poslední ztráty paketu
   5. Zachovává mnoho klíčových vlastností standartního TCP: spravedlnost, schopnost reakce, vztah k vyrovnávací paměti
5. BIC-TCP
   1. Výchozí algoritmus v Linuxovém jádru (2.6.8 a výš)
   2. Používá algoritmus binárního vyhledávání (binary-search) pro aktualizaci *cwnd*
   3. 4 fáze: (1) reakce na ztrátu paketů, (2) aditivní zbýšení (3) binární vyhledávání pro *cwnd* update (4) maximální sondování
   4. Přestože je algoritmus škálovatelný, spravedlivý a stabilní, jeho growth funkce je považována stále za agresivní pro TCP, obzvláště na pomalých sítích nebo krátkých RTT
6. CUBIC-TCP
   1. Novější verze BIC, která využívá kubickou funkci za účelem jednodušší analýzy protokolu. Počet jednotlivých fází je redukován.

## Rozšíření TCP s podporou IP

1. QuickStart (QS)
   1. Předpoklad: slow-start fázi nelze provést bez interakce s nižšími síťovými vrstvami
   2. Návrh: 4-bytové možnosti voleb v IP hlavičce – QS TTL a Initial Rate pole.
   3. Odesílatel, který chce použít QS, nastaví QS TTL na libovolnou hodnotu (dostatečně vysokou) a Initial Rate na požadovanou hodnotu, na které chceme začít odesílat data, a odešle SYN paket.
   4. Každý router na síťové cestě, který podporuje QS, snižuje QS TTL o jedničku a snižuje i Initial Rate, pokud je to třeba
   5. Příjemce odešle QS TTL a Initial Rate v SYN/ACK paketu odesílateli
   6. Odesílatel ví, jestli všechny routery na cestě podporují QS (na základě porovnání QS TTL a TTL)
   7. Odesílatel odešle *cwnd* a začne používat svůj mechanismus pro kontrolu zahlcení (např. AIMD)
   8. Algoritmus vyžaduje změny v IP vrstvě.
2. E-TCP
   1. Obsahuje tzv. Early Congestion Notification (ECN), congesition = zahlcení
      1. Komponenta Advanced Queue Managementu (pokročilého manažeru fronty)
      2. Bit nastavený routerem, když se blíží zahlcení linky/vyrovnávací paměti/fronty
      3. ECN flag musí být zrcadlen příjemcem
      4. Pokud je ECN = 1, TCP by mělo zareagovat stejným způsobem, jako na ztrátu paketů
      5. Vyžaduje administraci routerů k tomu, aby nastavil AQM/ECN
   2. Navrhuje zrcadlit ECN bit jen jednou (poprvé)
   3. Zmrazí *cwnd*, když je od příjemce přijat ACK příznak, který má nastaven ECN bit
   4. Vyžaduje změnu chování příjemce k ECN bitu
3. Fast AQM Scalable TCP (FAST)
   1. Využívá end-to-end spojení, ECN a ztrátu paketů pro detekci/vyvarování se zahlcení – pokud je v routeru ve frontě příliš málo paketů, je zvýšena intenzita zasílání
   2. Rozdíly oproti TCP Vegas:
      1. TCP Vegas dělá změny v rychlosti, které mají fixní velikost. Změny jsou nezávislé na tom, jak rozdílná je aktuální rychlost oproti chtěné rychlosti
      2. FAST TCP provádí větší kroky, když je systém více vyveden z rovnováhy a menší kroky, když je systém blízko k rovnováze
      3. Pokud je v síti k dispozici ECN, FAST TCP může být rozšířeno o využití ECN značení pro lepší vypořádání se se zpožděním a ztrátami paketů

## Přístupy odlišné od TCP

1. Tsunami
   1. Zaplaví síť daty
   2. TCP out-of-bound kanál pro vytvoření spojení:
      1. Vyjednávání o parametrech připojení
      2. Požadavky na znovuodesílání – místo ACK využívá příznak NACK
      3. Vyjednávání o ukončení připojení
   3. UDP kanál pro přenos dat
      1. MIMD kontrola zahlcení
      2. Velké možnosti konfigurace a přizpůsobení
2. Reliable Blast UDP – RBUDP
   1. Podobné jako tsunami – TCP out-of-bound kanál pro kontrolu spojení, UDP pro přenos dat
   2. Navrženo pro disk-to-disk přenosy = přenosy, kde celá přenášená data mohou být do odesílatelovi paměti
   3. Data se odesílají frekvencí, kterou definuje uživatel (podle síťové kapacity)
3. eXplicit Control Protocol (XPC) – využívá speciální **hlavičku zahlcení** obsahující zpětnou vazbu od routerů o paketech na routeru; *cwnd = cwnd* + *feedback*
4. SCTP
   1. Multi-stream, multi-homed přenos (koncový uzel má několik IP adres)
   2. SCTP je zaměřen na zprávy, jako např. UDP, zajišťuje tedy spolehlivost, přenos zpráv ve správném pořadí, a to s kontrolou zahlcení – jako TCP
5. DCCP (linux) – nespolehlivý protokol (UDP) s kontrolou zahlcení kompatibilní s TCP
6. STP
   1. Založeno na CTS/RTS
   2. Jednoduchý protokol navržený pro jednoduchou implementaci v HW
   3. Nemá žádný sofistikovaný mechanismus kontroly zahlcení.
7. Spolehlivý UDP
   1. Spolehlivé doručení paketů se zachováním pořadí
   2. Původně navrženo pro IP telefonii
   3. Parametry spojení mohou být nastaveny pro každé spojení zvlášť

Agresivní protokoly jsou také používány na soukromých a specializovaných sítích (CESNET)

# Peer-to-peer (P2P) sítě

## Client-Server vs. Peer-to-peer

Distribuované aplikace:

* Distribuované aplikace se skládají z vícero SW modulů, které se nachází na různých počítačích. Moduly spolu navzájem interagují přes komunikační síť a propojují různé počítače.
* Síť je používána pro synchronizaci a komunikaci mezi moduly.
* Více uživatelů může využívat aplikace na různých počítačích současně.
* K vybudování distribuované aplikace je třeba rozhodnout:
  + Jak umístit SW moduly na různé počítače v síti
  + Jak každý SW modul najde ostatní moduly, se kterými potřebuje komunikovat

### Client-Server

Client-Server systém se skládá z dvou typů softwarových modulů:

**Serverový modul**

* Jedna centralizovaná instance (může ale být interně replikována do více strojů – kvůli škálování)
* Pasivně naslouchá na připojení od klientů
* Může obsluhovat více žádostí od klientů:
  + Sekvenčně
  + Současně (vícevláknové servery)
  + Pomocí dalších replikovaných serverů na jiných lokacích
* Čekající žádosti od uživatelů můžou být řazeny do fronty
* Servery jsou spolehlivé, často běží v datacentrech

**Modul Klient**

* Vícero distribuovaných instancí, často i kontrolovaných různými uživateli
* Aktivně iniciují připojení k serveru
* Mezi klienty není žádné přímé spojení
* Klienti potřebují vědět síťovou adresu a číslo portu serveru
* Dosažení služby je obvykle provedeno pomocí konfigurace klienta

### Peer-to-Peer

* P2P systém – mnoho identických SW modulů (peerů), které běží na různých počítačích
* Peery mezi sebou komunikují přímo
* Každý peer je zároveň server i klient (poskytuje služby ostatním peerům, požaduje služby od ostatních peerů)
* Na rozdíl od dedikovaných serverů mají peery tendenci být nespolehlivé
* Dosažení služby je komplikovanější, protože v síti existuje více serverů, které se průběžně objevují na různých lokacích a zase mizí
* Díky více serverům (zde peerům) jsou P2P sítě přirozeně škálovatelné
* P2P sítě mohou pracovat bez dedikovaných serverových zařízení

**Peer-to-peer (P2P)** systémy jsou distribuované systémy, které se skládají ze vzájemně propojených uzlů. Uzly jsou schopny se sami organizovat do síťových topologií za účelem sdílení zdrojů, jako jsou např. uživatelská data, CPU smyčky, úložiště, šířka pásma. Jsou schopny se přizpůsobit výpadkům, zatím co udržují přijatelné spojení a výkon bez potřeby zásahu nebo podpory globálních centralizovaných serverů.

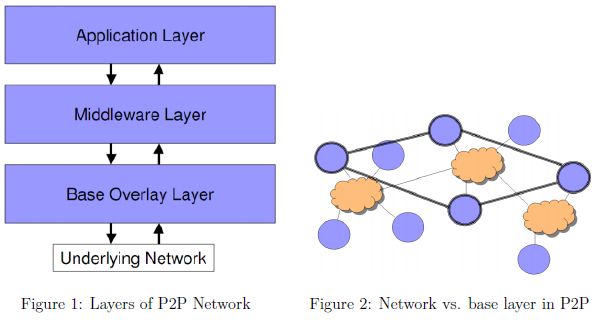
**P2P vlastnosti**

1. **Symetric role** – Každý uzel v síti se typicky chová jako server i klient, nicméně ve spoustě návrhů je tato vlastnost uvolněna při použití speciálních rolí (super peer, relay peer)
2. **Scalability** – P2P systém se může škálovat do tisíců uzlů, P2P protokoly nemůžou vyžadovat komunikaci „každý s každým“
3. **Distributed control (Decentralization)** – ideálně by neměla existovat žádná centralizovaná struktura mezi uzly
4. **Heterogeneity** – P2P systém je (obvykle) heterogení – uzly mívají různou kapacitu HW
5. **Dynamism** – dynamika, topologie P2P sítí se může rychle měnit s tím, jak se připojují nebo odpojují uzly do sítě
6. **Resource sharing** – sdílení zdrojů. Každý peer se dělí o systémové zdroje (výpočetní síla, data, šířka pásma atd.)
7. **Self-organization** – organizace P2P systému se časem zvyšuje na základě lokálních informací o síti

### Client-Server vs. Peer-to-peer porovnání

* **Snadnost vývoje**
  + C-S je stabilnější a známější než P2P
  + C-S ukazuje jednoduché interakční vzory pro klienty a server, kdežto P2P zahrnuje komplexnější interakční vzory mezi peery
* **Ovladatelnost**
  + Centralizovaný server v C-S prostředí je jednodušší na údržbu oproti udržování několika distribuovaných peerů v P2P systému
* **Škálovatelnost**
  + C-S škálovatelnost je limitována neměnným serverovým HW, nicméně škálovatelnosti lze dosáhnout rozložením zátěže mezi více serveru (zvýšené finanční náklady)
  + P2P je škálovatelné samo o sobě – spolu s tím, jak se zvyšuje počet peeru v síti, tak se zvyšuje i „serverová“ kapacita
* **Bezpečnost**
  + Zodpovědnost za bezpečnost v C-S leží na serveru, který je centrálně hostován v zabezpečeném prostředí (datacentrum). Jeden server je ale jednoduché napadnout.
  + Zodpovědnost za bezpečnost v P2P systému je distribuována mezi peery s různými doménami. Některé z nich můžou však být ohroženy. Není lehké zabezpečit P2P systém.
* **Spolehlivost**
  + Spolehlivost C-S systému je dosažena tím, že vytvořeny redundantní servery a ty jsou někdy uloženy na různých lokacích (zvýšení nákladů)
  + V P2P systémech je odolnost systému bezplatná, protože více peerů může poskytovat stejné služby.

## Obecná P2P architektura



**Base Overlay Layer**

* Base overlay layer je vrstva, která je zodpovědná za: objevování nových peerů, udržování P2P overlay sítě (překryvná síť), přeposílání zpráv mezi peery
* Overlay síť je virtuální síť, která leží na „fyzické“ síti (např. TCP/IP)
* „Kabely“ overlay sítě jsou implementovány pomocí základních síťových vybavení (TCP připojení, UDP zprávy)
* Overlay síťová vzdálenost je měřena v počtech hopů mezi peery
* Peery, které jsou ve fyzické síti vzdálené, mohou být v overlay sít sousedé a naopak.
* Výkon P2P systému je ovlivněn strukturou overlay sítě

**Middleware Layer**

* P2P systém je aplikace na vrstvě L7
* Middleware vrstva usnadňuje vývoj P2P aplikací tím, že skrývá overlay problémy a problémy s dosažením služeb
* Poskytuje přístup ke službám/zdrojům, které jsou poskytovány peery a může být zodpovědná za funkce, jako jsou:
  + Bezpečnost: kontrola přístupu ke službám a zdrojům
  + Dosažení služeb a zdrojů: vyhledávání a indexování služeb a zdrojů, které jsou rozděleny mezi peery
  + Skupiny peerů: koordinace peerů, které poskytují nebo spotřebovávají konkrétní služby/zdroje.
* Např. JXTA (Java P2P platforma), BOINC, P2, Windows P2P Networking

**Application Layer**

* Middleware služby mohou být využity pro tvorbu celých aplikací (sdílení souborů – Napster, Gnutella; routovací protokoly; instant messaging; Skype atd.)

### Overlays and Peer Discovery

* P2P síť je typicky „virtuální“ síť, která je postavena na existující síti – např. Internetu. Překrytí této sítě je využito pro indexování a dosažení peerů a dělá P2P síť nezávislou na fyzické topologii sítě. Obsah je obvykle vyměňován přímo po základní IP síti.
* Nový peer potřebuje nalézt alespoň jeden existující peer, aby se k P2P síti mohl přidat – potřebuje informaci o lokaci sítě: IP adresu, port pro naslouchání atd.
* Pokud není ihned nalezen žádný peer, nový peer buď pasivně čeká na nové účastníky v P2P síti, nebo proaktivně nové účastníky hledá.
* Ve velkých sítích, jako je Internet, je obtížné lokalizovat existující peery.

**Statická konfigurace**

* Každý peer má přednastaven seznam síťových lokací (IP adresy a čísla portů) všech ostatních peerů v systému.
* Na začátku (a pokud možno tak periodicky) se každý peer pokouší připojit k nějakému jinému peeru na svém seznamu, z nichž některé můžou běžet.
* Kvůli manuální konfiguraci je tato metoda vhodná pouze pro P2P sítě s malým počtem peerů, které se nemění často.
* Může být použita pro počáteční kontakt malého počtu „známých“ peerů, které budou určitě připojeny.

**Centralized direktory**

* Každý peer má přednastavenou síťovou lokaci centralizovaného serveru
* Každý peer na začátku kontaktuje server (pokud možno opakovaně) za účelem získání aktuálního seznamu aktivních peerů a za účelem informování serveru, že je tento peer aktivní.
* Většina následných komunikací už server obchází, využívá P2P překryvnou vrstvu ke směrování zpráv.
* Peery se mohou odpojit:
  + **Čistě –** v rámci ukončení peer kontaktuje server, aby ho odstranil ze seznamu aktivních peerů
  + **Bez varování** – selhání sítě, energie, peeru. Tím se stane seznamu aktivních peerů na serveru neaktuální (je nutné aktuálnost seznamu stále kontrolovat)
* Jeden peer se obvykle potřebuje připojit jen k několika málo dalším peerům na overlay síti. Ostatní členové mohou být dosaženi pomocí techniky *member propagation*
* Centralizovaný server může být tzv. single point of failure – „slabé místo“ v síti

**Member Propagation techniky s objevením počátečního člena**

* Obvykle není nutné snažit se objevit všechny členy v síti – dostačující je jen nějaká podmnožina
* Jakmile dojde k nalezení alespoň jednoho existujícího peeru, lze z něj získat informace o všech dalších peerech v P2P síti
  + Pokud si každý peer udržuje svůj kompletní seznam členů
  + Nebo pokud každý peer může spravovat částečný seznam členů

### Overlay Network topologie

* Prostřední peery v overlay síti přeposílají zprávy mezi nepřímo propojenými peery
* Overlay topologie významně ovlivňuje výkon P2P systému
* Efektivitu overlay sítě výrazně ovlivňují dvě klíčové vlastnosti:
  + **Diameter(Průměr):** nejdelší vzdálenost mezi dvěma peery, je počítáno TTL (jiné než u paketů). Diameter by měl být co nejmenší.
  + **Average Degree:** průměrný počet spojení na jeden peer (vysoký AD zvyšuje zátež zprávami, ale vylepšuje toleranci vůči chybám), měl by být držen na střední hodnotě.

**Random Mesh (náhodné „oko“)**

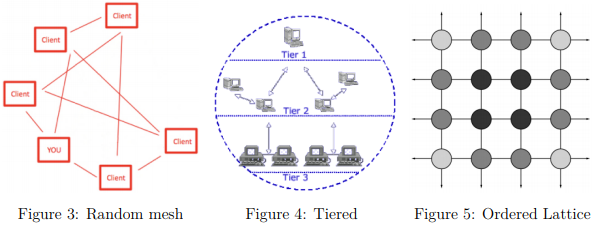
* Každý peer objeví určitý počet dalších peerů a pokusí se k nim připojit
* Tento přístup by měl vést k náhodné struktuře s jednotným stupněm
* Peery, které jsou vzdálené na základní síti, mohou být na overlay síti sousedé => připojení k peerům, které mají nejmenší odezvu
* Random mesh je vhodné pro spojování velkého množství peerů se stejnými zdroji a schopností připojení
* Lze jednoduše využít mechanismus search message flooding za účelem objevení zdrojů nebo služeb peerů, ale je při tom vygenerován velký síťový provoz

**Tiered Structure (vrstvená struktura)**

* Peery jsou seřazeny do vrstev stromu, které záleží na zdrojích, které nabízí a na schopnosti připojení.
* **Tier 0** je zakládající vrstva obsahující spolehlivé peery s adekvátními zdroji a kapacitou pro přeposílání zpráv
* V každé vrstvě je každý peer propojen s určitým počtem peerů z nižší vrstvy a přeposílá zprávy nahoru a dolů.
* Existují peery, které mají slabé nebo dokonce žádné zdroje – „poorle-resources leaf peers“. Jsou propojeny jen se svým „super-peerem“ a nepřeposílají zprávy ostatních peerů. Jsou také vynechány z procesu objevování peerů.
* Systém musí být schopný zotavit se ze situace, kdy peer náhle opustí síť a přeruší tak strom vrstev
* Hierarchie může být optimalizována tak, aby následovala strukturu základní sítě (např. P2P video streaming)

**Ordered Lattice (uspořádaná mřížka)**

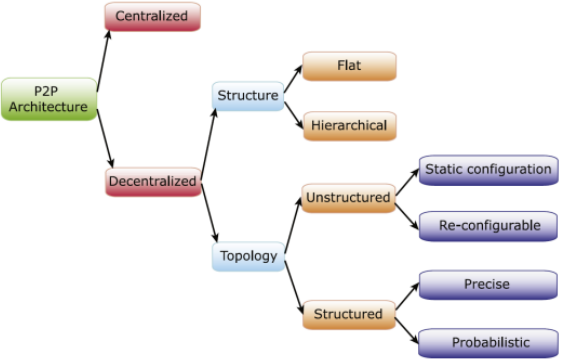
* Uspořádání peerů dvourozměrné mřížce, peery samy sebe organizují do obdélníkové struktury: každý uzel spravuje přímá spojení ke čtyřem dalším sousedům (kromě okrajových peerů).
* Mřížka může být rozšířena do *n* rozměrů
* Zprávy jsou směrovány paralelně k osám mřížky
* Přidávání a odebírání peerů musí být obslouženo za běhu, může rozbít strukturu mřížky – vkládání a odstraňování uzlů znamená, že různé řádky/sloupce mají různý počet členů
* Souřadnice peeru ve vícerozměrné mřížce mohou být použity jako klíč k lokalizaci zdrojů v tzv. „content adressable networks“ (CAN) – někdy také označovaných jako „Distributed Hash Table“ (DHT)



### Service/Resource Discovery – objevování služeb/zdrojů

* Peer musí nabízet své služby, aby umožnil jejich zjištění a následné použití ostatními peery
* Zjišťování služeb je samo o sobě službou
  + **Centralized** – server je požádán o lokaci služby (Napster, UDDI pro webové služby)
  + **Čistý P2P** – žádost je rozeslána do sítě nebo zahashována přes peer flooding, overlay multicast, CAN/DHT
* Když nalezne search message požadovanou službu na peeru, lokace serveru je odeslána zpět k původci
* Skutečné service messages jsou buď směrovány přes overlay vrstvu, nebo přímo pomocí základní vrstvy sítě
* Možnost optimalizace pomocí cáchování nabízených služeb a dat

## Taxonomie P2P systému



### Centralizované P2P systémy

* Kombinuje výhody centralizovaného (C-S) a decentralizovaného systému.
* Výhody centralizovaného přístupu: Existuje jeden nebo více centrálních serverů, které pomáhají peerům najít jejich požadované zdroje nebo se chovají jako plánovače úloh, aby koordinovaly akce mezi peery
* Peer odešle zprávu na centrální server, aby rozhodl adresy peerů, které obsahují požadované zdroje
* Stejně jako u decentralizovaných systému, jakmile peer získá požadované informace/data, může komunikovat s dalšími peery přímo, tzn. bez dalšího přístupu na server
* Nevýhody: citlivé na škodlivé útoky; SPoF (Single Point of Failure – slabé místo. Když zde systém selže, tak selže i celý systém); nedostačující pro velké množství peerů (snižování výkonu); nedostatek škálovatelnosti a robustnosti
* Příklady: Vědecké výpočty – SETI@home, BOINC; sdílení digitálního obsahu – Napster; ostatní – Jabber (Instant Messaging), StarCraft (počítačová hra)

### Decentralizované P2P systémy

* Peery mají stejná práva a zodpovědnosti
* Každý peer vidí pouze část P2P sítě a poskytuje data/služby, které mohou být relevantní pouze pro některé peery
* Kritickými a zásadními operacemi jsou lokalizace peerů, nabídka služeb či dat, to vše v dostatečné rychlosti
* Výhody: imunní k SPoF, (obvykle) vysoký výkon, škálovatelnost, robustnost a další požadované vlastnosti
* Příklady: Gnutella, Crescendo, PAST, FreeNet, Canon atd.

### Dva rozměry v návrhu decentralizovaných P2P systémů

1. Ploché (jednoúrovňové) vs. Hierarchické (víceúrovňové) síťové struktury
   1. **Flat structure (= plochá/nízká struktura)** – funkcionalita a zátěž jsou rovnoměrně rozděleny na zúčastněné uzly.
   2. **Hierarchická struktura** – více vrstev směrovacích struktur, např. národní úroveň (propojující státy), státní úroveň (propojující univerzity), univerzitní úroveň (propojující fakulty) atd.
   3. nabízí určité výhody (izolace chyb a bezpečnosti, efektivní caching a využití šířky pásma, hierarchické úložiště atd.)
2. strukturovaná vs. nestrukturovaná topologie
   1. **Nestrukturované P2P systémy**
      1. Každý peer je zodpovědný za svoje vlastní data a udržuje si informace o svých sousedech, kterým může přeposílat požadavky
      2. Obtížné přesně předpovědět, které peery spravují zpracovávaná data
      3. Neexistuje žádná záruka odpovědi od konkrétního peeru (pokud není prohledána celá síť)
      4. Neexistuje žádná záruka času odpovědi (kromě nejhoršího případu, kdy je prohledána celá síť)
   2. **Strukturované P2P systémy**
      1. Umístění dat je pod kontrolou určité předdefinované strategie (obecně distribuovaná hash tabulka – DHT)
      2. Mezi daty a peery existuje mapování
      3. Může poskytnout záruku na cenu vyhledávání
      4. Nicméně je to obvykle za cenu správy dodatečných údajů

### Hybridní P2P systémy

* Kombinují výhody centralizovaného a decentralizovaného přístupu k P2P systémům.
  + Rychlé a spolehlivé nalezení zdrojů, ale s omezením škálovatelnosti
  + Škálovatelnost, ale naopak je třeba delší čas pro nalezení zdrojů
* Aby šlo udržovat škálovatelnost, nejsou žádné centrální servery, jsou zde ale super peery – uzly, které jsou vybrány, aby se chovali pro ostatní jako servery
* Lokalizace zdrojů oběma přístupy (centralizovaný, decentralizovaný)

## Routování zpráv a žádostí v P2P sítích

* Jedna z klíčových operací v P2P systémech
* Odlišné algoritmy oproti L3 routování. V L3 vrstvě hledáme cestu, route, pro pakety, na vrstvě L7 (P2P) hledáme cestu k uzlu, který obsahuje požadovaná data.
* Návrh routing protokolů je jeden z nejvíce zkoumaných problémů.
* Rozdíl v množství ukládaných metadat
  + **S metadaty** – Abychom mohli lokalizovat data, měl by každý peer být schopen přeposílat požadavky na podmnožinu sousedních peerů, které jsou blíže k cílové destinaci než ostatní peery (např. Napster).
  + **Bez metadat** – lokalizace informací probíhá zaplavováním/vysíláním požadavků do sítě (např. Gnutella).

**Metriky efektivity a účinnosti schématu routování:**

* Úložiště – každý peer může potřebovat založit nějaké úložiště pro metadata. Je náročnější udržovat metadata aktuální.
* Efektivita – schopnost lokalizovat zdroje rychle. Metrikou zde je response time – čas odpovědi.
* Použitelnost – reflektuje snadnost užití a typy požadavků, které mohou být podporovány, např. v závislosti na spravovaných metadatech může jeden systém podporovat komplexní požadavky, kdežto jiný může splňovat jen velmi konkrétní poždavky
* Pokrytí – jestliže prohledávaný prostor obsahuje odpověď, schéma s vyšším pokrytím bude určitě užitečnější.
* Škálovatelnost – je důležité vytvořit schéma tak, aby bylo užitečné v prostředích s velkým měřítkem. Metrikou škálovatelnosti může pak být např. počet zpráv, které musí být směrovány za účelem lokalizování informace.

## Routování v nestrukturovaných P2P sítích

* Každý peer si ukládá vlastní datové objekty a udržuje své vlastní odkazy na sousedy.
* Když se chce uzel A připojit do systému, kontaktuje existující uzel B a okopíruje jeho spojení, aby si vytvořil své vlastní
* Žádný peer nemá globální ponětí o umístění svých dat => pro požadavky se využívají zaplavovací techniky (TTL se připojuje ke každému požadavku)
* Některé routovací strategie: BFS (např. Gnutella, DFS (např. FreeNet, Heuristicky založené strategie

### Heuristicky založené routovací strategie

**Iterative Deepening**

* Požadavek je iniciován sekvencí několika tradiční BFS vyhledávání a postupným rozšířením vyhledávacího poloměru. Proces vyhledáván končí, když je dosaženo maximální hloubky, nebo výsledky požadavku splňují požadavky uživatele.
* Pravidla systému musí být poskytnuta, aby specifikovala posloupnost jednotlivých hloubek, ve kterých se vyhledávací iterace provádí (*P = D1, D2, …, Dn,* kde *D1 < D2 < … < Dn*).
* Na základě těchto požadavků odešle nejdříve zdrojový uzel dotaz do sítě pomocí BFS vyhledávání hloubky *D1.* Pokud získaná odpověď vyhovuje požadavkům uživatele, je dotaz ukončen. V opačném případě odešle zdrojový uzel další dotaz do sítě (se stejným query ID) pmocí BFS vyhledávání, ale tentokrát hloubky *D2*.
  + Uzly, které jsou méně jak *D1*-hopů od zdrojového uzlu, nedělají nic, jen přeposílají query dotaz svým sousedům.
  + Další uzly zpracují dotaz stejně, jako při první iteraci.
* Podobně to probíhá i pro *D3, D4*, atd. Pokud není dotaz vyřešen ani při dosažení hloubky *Dn*, proces vyhledávání je ukončen.

**Directed BFS** – Každý uzel posílá dotazy jen na část svých sousedů. Každý uzel si udržuje některé statistiky o svých sousedech, např. počet dříve zodpovězených dotazů, počet získaných výsledků atd. Uzly jsou vybírány na základě těchto statistik.

* Výhoda: počet query dotazů v síti je tímto znatelně redukován (v porovnání se standartním BFS)
* Nevýhoda: uzly jsou povinné udržovat si statistiky, které mohou být někdy zavádějící.

**Inteligentní vyhledávání** – každý peer hodnotí své sousedy na základě relevantnosti odpovědí na dotazy. Toto hodnocení poskytuje přesnější výsledky hodnocení peeru než Directed BFS.

**Local Indices Search (prohledávání lokálních indexů)**

* Každý uzel si vytváří a udržuje indexy na lokálních datech i na datech svých sousedů, kteří jsou v poloměru *k* hopů. Pokud je *k =* 0: podobné jako BFS vyhledávání (indexy pouze na lokálních datech). Vrácený výsledek takového uzlu je stejný, jako výsledek, který by byl vrácen po zpracování dotazu na všech uzlech v poloměru *k* hopů od uzlu.
* Dotazy jsou zpracovány na základě globálních pravidel P – specifikují seznam hloubek. Uzly na specifické hloubce zpracovávají dotaz, ostatní jen přeposílají dotaz svým sousedům.
* Výhoda: Cena zpracování je snížena – dotaz je zpracováván méně uzly
* Nevýhoda: Vyšší cena úložiště a aktualizace (ukládání indexů). Hrozí také nekonzistence indexů (kvůli dynamice sítě)

**Random Walk**

* Náhodně vybíráme souseda, kterému přepošleme dotaz, a to tak dlouho, dokud nenajdeme hledané výsledky, nebo expiruje TTL.
* Kvůli dlouhému času zpracování existuje tzv. *k-walker Random Walk Algorithm*
  + Odesílá dotaz svým náhodně vybraným sousedům – v první iteraci
  + V dalších iteracích: jakmile uzel obdrží dotaz (tzv. walker), vybere si pouze jednoho souseda.
  + Počet zpráv (navštívených uzlů) se lineárně zvyšuje v porovnání s 1-walker algoritmem.

**Random Breadth First Search (RBFS)**

* Podobné jako k-walker Random Walk
* Iniciátor dotazu náhodně vybere podmnožinu svých sousedů, kterým pošle dotaz, a každý z těchto sousedů náhodně vybere podmnožinu svých sousedů atd.
* Počet zpráv roste exponenciálně oproti 1-walker algoritmu

**Adaptive Probabilistic Search (Přizpůsobivé pravděpodobnostní vyhledávání)**

* Kombinuje *k-walker random search* a *probabilistic search*
* Každý peer obsahuje pravděpodobnost každého svého souseda s ohledem na každý objekt (vypočítáno z dřívějších výsledků) – dotazy jsou odeslány sousedům na základě pravděpodobností
* Dva přístupy, jak aktualizovat pravděpodobnost
  + Optimistický přístup – systém proaktivně zvyšuje pravděpodobnost vybraným (=dotazovaným) sousedům na vyhledávací cestě a snižuje jejich pravděpodobnost jen tehdy, když walker, který přes ně prochází, selže.
  + Pesimistický přístup – systém proaktivně snižuje pravděpodobnost vybraným (=dotazovaným) sousedům na vyhledávácí cestě a zvyšuje jejich pravděpodobnost jen tehdy, když walker, který přes ně prochází, skončí úspěchem.
* Swappin-APS – každý peer přepíná mezi optimistickým a pesimistickým přístupem (na základě hodnocení úspěšných walkerů pro každý objekt)
* Weighted-APS – zohledňuje lokaci objektů

**Interest-Based Shortcuts**

* Každý peer přidává dodatečné odkazy (zkratky zájmu – interest-based shortcuts), které spojují dva peery se stejnými zájmy na vrcholu existující vyhledávácí sítě, aby došlo ke zvýšení výkonu při vyhledávání.
* Když peer zpracovává dotaz, nejprve jsou využity interest-based shortcuts k přeposlání a zpracování dotazu. Pokud je nalezen nějaký výsledek, vyhledávání končí. V opačném případě pokračuje normální zpracování dotazu podle algoritmu, který je použit.
* *Konstrukce shortcuts*:
  + Když se peer připoj do sítě, nemá žádnou zkratku.
  + Poté, co úspěšné zpracuje dotaz, přidá iniciátor dotazu zkratku k peerům, které poskytly požadovanou odpověď na daný dotaz.
  + Každý peer si ukládá jen omezený počet zkratek, které jsou nejvíce využívané (kvůli omezení na využívání místa)

## Routování ve strukturovaných P2P sítích – intro

* Nestrukturované P2P sítě trpí problémem špatné efektivity vyhledávání
* Zúčastněné uzly ve strukturovaných P2P jsou povinny organizovat se do nějakých topologií (na rozdíl od nestrukturovaných P2P systémů) – jako je např. ring (Chord), vícerozměrná mřížka (CAN), mesh (Pastry, Tapestry), multiple list (Skip Graph, atd. – když se do systému přidá nový uzel, systém musí dodržovat striktní pravidla k nastavení jeho pozice
* Pokud výsledek dotazu existuje v systému, bude nalezen – navíc efektivním způsobem
* Nevýhoda: kvůli síťové topologii jsou i vysoké náklady na její udržování
* Na základě overlay síťové struktury můžeme strukturované P2P systémy klasifikovat takto:
  + **Systémy založené na Distributed Hash Table (DHT)** – Chord, CAN, Tapestry
  + **Systémy založené na Skip List** – Skip Graph, SkipNet
  + **Systémy založené na Tree** – P-Grid, P-Tree, BATON

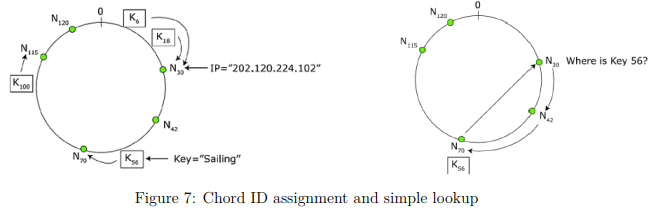
## P2P systémy založené na Distributed Hash Table (DHT)

### Chord

* Jeden z nejznámějších routovacích mechanismů ve strukturovaných P2P sítích
* Každý uzel v P2P síti si sám spravuje svou část globální hashovací tabulky
* Uložení/získání položky/položek znamená odeslání dotaz na uzel, který spravuje tu část tabulky, které náleží příslušný(é) hash(e)
* **Idea:** jednocestná konzistentní hashovací funkce namapuje každý uzel a data na *m-*bitové ID v jednorozměrném ID prostoru. Hashovací funkce využívá IP adresu uzlu k tomu, aby vygenerovala identifikátor pro uzel, a data (nebo klíč k datům), aby vygenerovala ID pro data. ID prostor musí být vybrán dostatečně veliký (pravděpodobnost přiřazení stejného ID různým uzlům by měla být zanedbatelná)
* **Detaily:** ID prostor je okruh čísel od 0 do 2m – 1. Systém přiřazuje klíč *k* prvnímu uzlu *n*, jehož ID je rovno nebo následuje číslo *k* v kruhovém prostoru – tzn. klíč *k* je přiřazen prvnímu uzlu prvnímu uzlu ve směru hodinových ručiček od *k.*

**Simple lookup algoritmus**: Každý uzel potřebuje znát jen své bezprostřední následníky. Když uzel obdrží žádost/dotaz:

* Zkontroluje své lokální úložiště, jestli nemá hledaná data
  + Pokud ano – je výsledek vrácen odesílateli dotazu
  + Pokud ne, uzel přepošle dotaz sousedu, který je jeho bezprostřední následník
* Hledání je ukončeno, když je výsledek nalezen nebo ID bezprostředního následníka uzlu překračuje ID dotazovaných dat – tím pádem se výsledek nepodařilo najít.
* Složitost je *O(N)* (*N* = počet uzlů v systému)



**Scalable lookup algoritmus (škálovatelný vyhledávací alg.):** Místo udržování pouze jednoho bezprostředně následujícího uzlu si každý uzel udržuje tzv. *finger table* – tabulku skládající se z *m* následujících uzlů. Když uzel *n* obdrží dotaz, tak:

* Pokud daný uzel nemá požadovaná data, prohledá svou finger tabulku uzlů a hledá uzel *n‘* s nejvyšším ID, které vyhovuje podmínce *n.id* < *n‘.id* < *k.*
  + Pokud takový uzel existuje, uzel *n* požádá uzel *n‘*, aby našel klíč *k.*
  + V opačném případě se uzel zeptá svého přímého následníka, aby našel *k*.
* Vyhledávání končí stejným způsobem jako u Simple Lookup algoritmu.

Složitost je *O*(*logN*) (*N* – počet uzlů v systému).

**System construction:** Jakmile se uzel připojí k systému, musí udělat následující:

1. Najít svou pozici v kruhu (topologie) a získat data, za která je zodpovědný (na základě klíčů)
2. Inicializovat svou finger table
3. Aktualizovat finger tabulky ostatních uzlů, aby reflektovaly přítomnost nového uzlu

Když existující uzel opustím systém, nemusí dělat vůbec nic.

### Content Addressable Network (CAN)

* **Idea:** Routing systém postavený na virtuální *d*-dimensionální kartézské soustavě souřadnic. Systém rozdělí úložný prostor na několik odlišných zón, z nichž každá je přiřazena nějakému uzlu. Tyto uzly si ukládají svoje data výhradě do přirazené zóny. Systém využívá jednotnou hashovací funkci, která mapuje hodnoty datových klíčů na bod *p* v soustavě souřadnic.
* Vkládání dat:
  + Hodnota datového klíče je namapována na bod *p* v soustavě souřadnic
  + Uzel *n*, jehož zóna pokrývá *p*, je nalezen a kontaktován, aby si uložil novou položku
* Zpracování dotazu je podobné – pokud existuje výsledek, měl by být uložen v zóně, kterou pokrývá patřičný uzel
* Každý uzel si potřebuje udržovat informace o svých sousedních uzlech – tzn. uzlech, které pokrývají přiléhající zóny.
* Routování je založeno na jednoduchém „chamtivém“ přeposílacím algoritmu – v každém kroku je vybrán uzel, který má bližší souřadnice k hledané zóně.

**Konstrukce systému:** Když se do systému přidá nový uzel, je potřeba:

1. Najít libovolný uzel, který je již připojen k síti
2. Identifikovat zónu, která by mohla být rozdělena, a požádat vlastníka této zóny o její rozdělení na dvě části – původní uzel bude udržovat jednu část, nový uzel začne udržovat druhou část
3. Sestavit vlastní routing table a aktualizovat routing tables sousedů

Když síť opustí existující uzel, musí požádat své sousedy, aby sloučili zóny do jedné.

### Pastry (=“pečivo“)

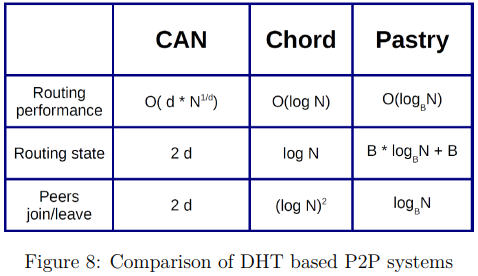
* Routovací systém založený na **PRR trees** – PRR = Plaxton, Rajaraman a Richa
* ID uzlu je *m-*bitové číslo rozdělené do posloupnosti čísel, které mají základ 2*b* (*b* – konfigurační parametr) – např. 128-bitový identifikátor je rozdělen na 32 čtyřbitových čísel (*b* = 4, základ = 16 => hexadecimální posloupnost čísel)
* Data jsou uložena na uzlu s ID, které sdílí *nejdelší prefix s datovým ID*
* V každém kroku routování je vybrán soused, která má delší společný prefix s cílovým uzlem (delší o 1 číslo, tzn. o *b* bitů)
* Složitost routování je 
* Každý peer má routovací tabulku, aby mohl směrovat zprávy
  + Tabulka má pevný počet úrovní  a na každé úrovni je pevný počet záznamů (= 2*b* – 1)
  + ID řádku = délka prefixu společného s cílovým uzlem
  + ID sloupce = další možný krok
* **Detaily:** Kromě routovací tabulky má každý uzel také nastaven *leaf set* (leaf=list)*,* který obsahuje seznam ukazatelů na uzly, které jsou numericky nejblíže k určitému uzlu a slouží jako spádové uzly, když v routovací tabulce není nalezen uzel s delším prefixem. Velikost leaf setu je typicky 2b – polovina tabulky obsahuje klíče menší, než jsou klíče příslušného uzlu, polovina tabulky obsahuje klíče větší.
* **Routování v Pastry:**
  + Pokud je vyhledávaný uzel v leaf set příslušného uzlu, je na základě toho provedeno routování. V opačném případě je poslána zpráva na uzel, který má nejdelší společný prefix s klíčem
  + Pokud je routovací tabulka prázdná nebo odkazovaný uzel není dostupný, je přeposlána zpráva na:
    - Souseda majícího stejně dlouhý společný prefix se stejným klíčem, jako má uzel, nebo
    - Na uzel, jehož ID je numericky blíže klíči, než je ID odesílajícího uzlu.

**Konstrukce systému**

* Když se do sítě přidá nový uzel (mající ID *X*), tak:
  + Kontaktuje libovolný připojený uzel *A* a mu pošle zprávu *join*(*X*)
  + Uzel *A* nasměruje zprávu *join*(*A*) na uzel *Z*, který je nejblíže klíči *X*
  + Uzel *X* obdrží leaf set uzlu *Z* a vyplní si svou routovací tabulku
  + Uzel *X* informuje uzly, které by si nový uzel měly vložit do svých routovacích tabulek.
* Když existující uzel opustí síť, tak:
  + Data, která spravoval, musí poslat sousedovi
  + Routovací tabulky se brzy automaticky aktualizují – uzel je nahrazen uzlem z leaf setu odpojeného uzlu (tedy jedním z jeho sousedů)

### Tapestry (=“gobelín“)

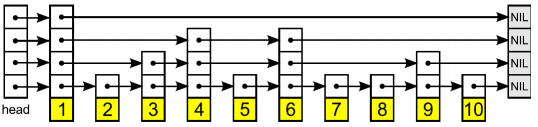
* P2P overlay routovací struktura založená na PRR Trees. Je velmi podobná Pastry.
* Hlavní rozdíl mezi Pastry a Tapestry – v Tapestry každý hop v routování rozšiřuje odpovídající sufix – v Pastry prefix.



## P2P systémy založené na Skip List

### Skip List struktura

* Skip List = datová struktura pro ukládání seřazeného listů položek s využitím hierarchie spojovaného seznamu
* Seznamy propojují rostoucí řídké posloupnosti položek
* Seznamy jsou sestaveny ve vrstvách:
  + Spodní vrstva (úroveň 0) je obyčejný seřazený spojovaný seznam
  + Každá vyšší vrstva je „expresní cestou“ (rychlou) pro seznamy z nižších úrovní, kde element ve vrstvě *i* se objeví ve vrstvě *i* + 1 s určitou fixní pravděpodobností *p* (obvykle je *p* = ½ nebo ¼ )



**Vyhledávání cílového elementu:**

* Začíná na horním elementu na vrcholu seznamu a postupuje se horizontálně, dokud aktuální element není větší nebo roven cílovému elementu
* Pokud je aktuální element roven cílovému, znamená to, že by cílový element nalezen.
* Pokud je aktuální element větší než cílový, procedura je po návratu na předchozí element opakována, a přitom dojde k sestupu na nižší vrstvu seznamu
* Předpokládaná složitost vyhledávání je *O*(*log n*)

### Skip Graph

* **Idea:** Routovací systém založený na Skip Lists. Čistý Skip Lists není ale vhodný, protože uzly vrchní úrovně mohou být přetíženy. Na rozdíl od čístého Skip List, který má jen jeden seznam na každé úrovni, Skip Graph má více listů na každé úrovní.
* Každý uzel se v každé úrovni nachází v nějakém seznamu.
  + Úroveň 1: propojíme všechny uzly se stejným prefixem délky 1(části prefixu, které se shodují, mají délku 1)
  + Úroveň 2: propojíme všechny uzly se stejným prefixem délky 2
* Systém kontroluje seznamy, ke kterým uzel náleží, pomocí tzv. **membership vector**(vektoru členství – vytvořen, když se uzel přidá do sítě)
  + Vektor jen definuje, ke kterému seznamu příslušný element náleží (seznamy jsou seřazeny podle datového klíče)
  + Vektor není unikátní!
  + Počet úrovní je *O*(*log N*)

**Vyhledávání cílového elementu:**

* Jakmile uzel vyšle dotaz, vyhledávání započne na nejvyšší úrovni daného uzlu
* Pokud existuje soused na stejné úrovni, který uchovává bližší hodnotu k vyhledávanému klíči, je mu v každém kroku přeposlán zpracovávaný dotaz
* V opačném případě uzel pokračuje ve vyhledávání na nižší úrovni
* Cílový uzel obsahující výsledek je nalezen, když vyhledávací proces dosáhne spodní úrovně.
* Složitost zpracování dotazu je *O*(*log n*)

**Konstrukce systému:** Když se nový uzel mající ID = X přidá do sítě, tak:

* Na základě svého membership vektoru *m*(*X*) se uzel X přidá do seznamu uzlů, jehož membership vektory mají stejný prefix jako *m*(*X*), ale v různých délkách.
* X se nejprve přidá do seznamu na úrovni 0 (uzly, které obsahují klíče nejblíže klíči uzlu X)
* Pro každý level *i* větší než 1 se X spojí s nejbližším uzlem Y, který má stejnou *i*-délku prefixu s uzlem X.

### SkipNet

* **Idea:** routovací systém velmi podobný Skip Graph. Namísto Skip Listů ale SkipNet organizuje uzly do rings (kruhů). Podobně jako u Skip Graph je Skip List organizován do úrovní, ve kterých jsou uzly seřazeny na základě datového klíče. Každý uzel má ve své routovací tabulce uloženy ukazatele na své sousedy.
  + Ukazatele na úrovni *h* ukazují na uzly, které jsou zhruba 2*h* uzlů nalevo nebo napravo od daného uzlu
  + Všechny uzly jsou spojeny kořenovým kruhem (ringem), který se formuje na úrovni 0
* Routing/lookup mechanismus a konstrukce systému jsou velmi podobné, jako u Skip Graph.

## Tree based systémy

### P-Grid

* **Idea:** P-Grid je založen na virtuální binární stromové struktuře, ve které každý peer udržuje listy stromu (listy zde představují jednotlivé uzly). Systém každému peeru přiřadí ID, které představuje řetězec binárních bitů reprezentující cestu od kořene k listu. Každý peer je poté zodpovědný za všechny datové položky, jejichž prefix je stejný jako ID daného peeru.
* Za účelem tolerance chyb může být více peerům přiřazeno stejné ID.
* Za účelem správného routování si každý peer udržuje routovací tabulku.

**Routing/lookup mechanismus:**

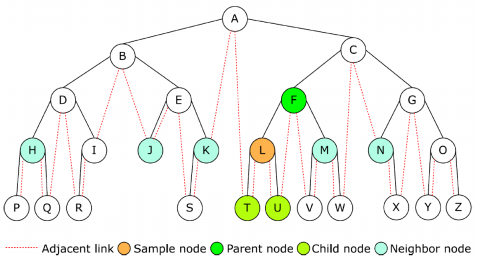
* Když peer *n* obdrží dotaz s klíčem *k*, zkontroluje, jestli jeho ID je prefix klíče *k*
  + Pokud ano, prohledá svoje lokální úložiště pro nalezení výsledku
  + Pokud ne, tak se peer *n* podívá do routovací tabulky, aby našel nejbližšího souseda, kterému přepošle dotaz.
* Maximální počet vyhledávacích kroků je omezen výškou stromu – složitost vyhledávání je *O*(*log*2 *N*).

### P-Tree

* **Idea:** v P-Gridu nelze garantovat vyvážení stromové struktury. P-Tree je založen na vyváženém B+-Tree vybudovaném na vrcholu „Chord ring“ (chord = struna, akord)
* Každý peer si udržuje tzv. *Chord uzel*, který je listem v B+ stromové struktuře, dále tzv. *semi-independent B+-Tree* (polo-nezávislý), který je tou částí plně nezávislého B+-Stromu, kterou vidí peery.
  + Plně nezávislý B+-Tree na peeru je takový B+-Tree, kde jsou hodnoty uložené na peeru považovány za nejnižší hodnoty v rámci Chord ring
  + Polo-nezávislý B+-Tree obsahuje všechny uzly v nejlevější cestě od kořene k listu z příslušného plně nezávislého B+-Tree.
  + Kvůli jednodušší údržbě se můžou rozsahy B+-Stromů překrývat

### Baton

* Ve většině stromově založených struktur platí, že stromy jsou jen virtuální, interní uzly nepřenáší data, uzly obsahují peery. V Baton platí, že také interní uzly obsahují peery.
* **Idea:** V porovnání se standartními stromově založenými strukturami Baton nabízí dvě hlavní funkce:
  + Data jsou seřazena v obou listech i v obou interních uzlech.
  + Kromě odkazů rodičů a potomků, uzly v BATON sítích mají také příslušné odkazy a odkazy na sousedy
    - Příslušný odkaz je použit ke spojení uzlu s uzlem, který spravuje sousední rozsah hodnot (sousedící s rozsahem, který uzel udržuje)
    - Odkaz na souseda je využit pro propojení uzlu s jeho sousedy (na stejné úrovni ve stromové struktuře), kteří mají vzdálenost 2*i*, *i* >= 0, od uzlu.
    - Účelem těchto odkazů je vyvarování se problému *bottleneck*, tedy „zúženého hrdla, cesty“, v kořenu celé stromové struktury, když dochází ke zpracování dotazu.



**Detaily vyhledávání:** Když peer X obdrží dotaz:

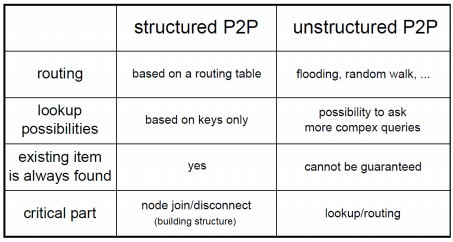
* Pokud vyhledávaný klíč spadá do rozsahu hodnot spravovaných peerem X, peer odpoví.
* V opačném případě peer přepošle dotaz na nejvzdálenějšího souseda, který je blíže ale ne příliš blízko k vyhledávanému klíči.
* Pokud takový soused neexistuje, peer X přepošle dotaz buď na potomka (pokud existuje), nebo na příslušný uzel ve směru vyhledávání.

## Routování v hybridních P2P sítích

Hybridní P2P systémy organizují peery do hierarchické sítě: silné peery (super-peers, supernodes) leží ve vyšších úrovních, obecné peery (také nazývané klientské peery) leží v nižších úrovních. Každý obecný peer náleží nějakému superuzlu a není propojený s žádným obecným peerem, který nenáleží stejnému superuzlu.

Obecné routovací schéma v hybridních P2P sítích:

* Klientský peer pošle dotaz na svůj superuzel
* Superuzel prohledá svůj adresář, aby rozhodl, který klientský peer nebo superuzel má požadovanou odpověď
* Dotaz je poslán na superuzel, který by mohl mít hledanou odpověď (využívá svůj adresář všech klientských peerů)
* Je vrácena IP adresa klientského peeru, který má požadovanou odpověď
* Dotazovaný peer si vymění s daným peerem hledané zdroje



# Ad-hoc sítě, MANETy

## Bezdrátové Ad-hoc sítě

Bezdrátové sítě jsou obvykle založeny na mobilní infrastruktuře – území, které by mělo být pokryto radiovými službami rozdělenými do buněk. Každá buňka je pokryta základnou. Základny jsou připojeny ke kabelové páteřní síti. Mobilní uzly komunikují s těmito základnami bezdrátově, síťový provoz mezi různými mobilními entitami je přenášena jak přes základny, tak přes páteřní kabelovou síť.

Pokud ale není k dispozici žádná infrastruktura (málo času/peněz na to, aby byla nastavena), je snaha o vybudování sítě bez infrastruktury, jen za využití síťových schopností účastníků komunikace.

**Ad-hoc síť**

* Kolekce autonomních uzlů, které mezi sebou komunikují tím, že vytvoří multihop radiovou síť a udržují připojení decentralizovaným způsobem.
  + Každý uzel funguje jako uzel i jako router
  + Kontrola sítě je distribuována mezi uzly
  + Síťová topologie je (obecně) dynamická – propojení mezi uzly se může v čase měnit, jak se uzly připojují nebo odpojují ze sítě nebo pohybují (jedná se o uzly v bezdrátové síti, tedy uzlem může být např. mobilní telefon).

**Výhody:**

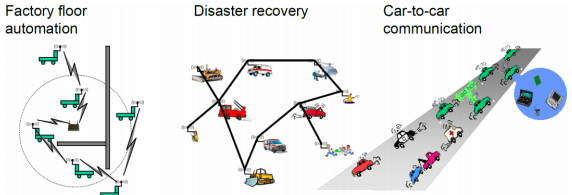
* Velmi rychlé budování sítě – není třeba zakládat kabelové spojení
* Pružnost – neexistuje žádný single point of failure (slabé místo)
* Mnohem více efektivnější, než mobilní sítě – každý uzel může komunikovat s jakýmkoliv jiným uzlem (někdy dokonce současně)

**Problémy/Výzvy:**

* Nedostatek centrálních entit, které pomáhají organizovat síť. Zúčastněné uzly se musí samy organizovat do sítě – schopnost sebeorganizace je nutnost.
* Omezený dosah bezdrátové komunikace – data musí být doručena přes síťové cesty, které zahrnují více uzlů – je vyžadován mechanismus pro dynamickou identifikaci síťových cest a jejich správu
* Mobilita členů sítě – síťové uzly se mohou pohybovat, kvalita sítě závisí na schopnosti sítě rychle se adaptovat na nové topologie – Mobile Ad-hoc Networks (MANETs)
* Kromě toho musí být vyřešeny tyto problémy:
  + *Medium access control* - žádná základna (base station) nemůže přiřazovat přenosové zdroje (ty musí být přiřazeny distribuovaným způsobem)
  + *Routing* – nalezení cesty z jednoho účastníka ke druhému
  + Musíme změnit vrstvy L2 a L3!

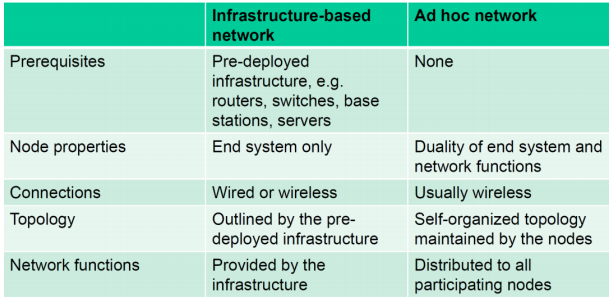
**Možné aplikace:**

* Nalezení prázdného parkovacího místa ve městě (bez dotázání se serveru), vyhnutí se dopravní zácpě atd. (=VANETs)
* Společné a distribuované výpočty



**Automobilové Ad-hoc sítě (Vehicular Ad-hoc Networks – VANETs)**

* Technologie, která k vytvoření sítě využívá pohybující se auta jako uzly/routery: Auta se mohou navzájem kontaktovat (pokud jsou v dosahu) a tím pádem mohou vytvořit síť s širokým dosahem.
* V porovnání s MANET sítí, kde se uzly pohybují náhodně, mají auta tendenci pohybovat se organizovaně (po silnicích)
* Navíc může být interakce se zařízeními na silnicích popsána poměrně přesně – můžeme použít specializovanější protokoly

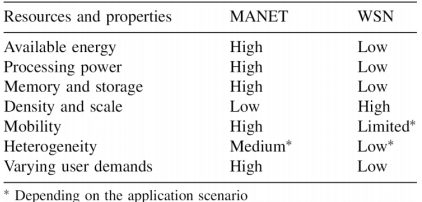


**Wireless Sensor Networks (WSNs – Bezdrátové senzorové sítě)**

* Namísto zaměření interakce na lidi zaměřují WSNs svou interakci na okolní prostředí
* Sítě jsou zakomponovány do okolního prostředí (náhodným nebo pravidelným způsobem)
* Uzly v takové sít jsou vybaveny snímači/ovladači k měření/ovlivňování okolního prostředí
* Uzly zpracovávají informace a komunikují bezdrátově
* Příklady užití:
  + *Nouzové situace –* např. shození senzorových uzlů do požárů, každý uzel měří teplotu a může tak vytvořit mapu teplot v požáru.
  + *Monitorování stanovišť –* např. při sledování stanovišť v divočině (stanoviště – místo, kde žije nějaký živočišný/rostlinný druh)
  + *Přesné zemědělství –* využití pesticidů, hnojení jen tam, kde je to potřeba
  + *Inteligentní budovy, mosty –* redukce plýtvání energií díky vhodné vlhkosti vzduchu, větrání, klimatizace atp. Potřebujeme přitom správné měření teploty a proudění vzduchu.
* **Sensor HW** – procesor (a paměť), rádiové vysílače/přijímače, baterie, senzory (světla, teploty, pohybu)
* **Sensor SW** – operace řízené na základě událostí (např. TinyOS)
* Využité síťové protokoly musí odebírat omezené množství energie a musí se chovat úsporně
* Nutná funkcionalita a omezení:
  + *Dostupná energie –* senzorové uzly mají baterie – ty se můžou vybít.
  + *Výkon při zpracování –* využití mikrokontrolery obvykle mají jen omezený výkon (kvůli omezení velikosti a spotřeby energie)
  + *Paměť a úložiště –* monitorování dostupné paměti obvykle souvisí s velikostí mikrokontroleru
  + *Šířka pásma a propustnost –* bezdrátové rádio vysílače jsou optimalizovány pro nízkoenergetické operace – poskytují aplikacím relativně malou šířku pásma
  + *Spolehlivost –* v závislosti na účelu aplikace se mohou požadavky na spolehlivost měnit
  + *Adresování –* běžné senzorové uzly nemají obvykle předprogramovanou globální unikátní adresu – síťové mechanismy musí buď adresu dynamicky alokovat, nebo dokonce zapomenou na využívání technik založených na adresách.
  + *Škálovatelnost* využívaných metod a algoritmů – primární omezení

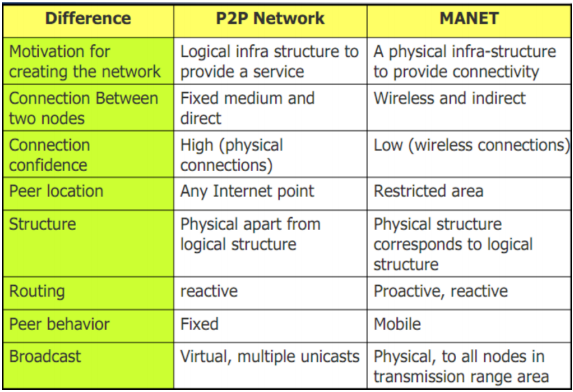
**MANETs (VANETs) vs. WSNs:**

* Podobnosti:
  + Obě sítě silně spoléhají na samoorganizační mechanismy (vztahy se sousedy a údržba síťové topologie)
  + Obě sítě musí zvládnout pracovat s omezeným množstvím energie v zařízeních – efektivita energie využitých algoritmů a metod má jeden z nejvyšších významů
  + Obě sítě často používají multi-hop komunikaci
* Rozdíly:



MANETs vs. P2P systémy:

* Bezdrátové Ad-hoc sítě mají také mnoho podobností s P2P systémy:
  + Stejné paradigma
  + Samoorganizační síť
  + Dynamická topologie
  + Zodpovědnost za routování dotazů v distribuovaném prostředí
  + Nedostatek řídicích a centralizujících jednotek
* Rozdíly:
  + MANET je spíše platforma pro P2P aplikace
  + Za účelem efektivního nasazení P2P aplikací v MANET síti, musí být exitující P2P řešení modifikováno, aby bralo v potaz specifika MANET sítě.



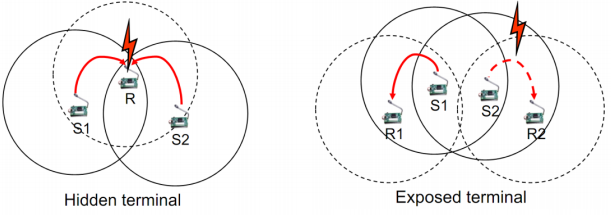
## Medium Access Control v Ad-hoc a Senzorových sítích

Medium Access Control (MAC)

* Zodpověnost za koordinaci přístupu uzlů ke sdíleným přenosovým médiím.
* Cílem je minimalizace kolizí
* Medium access (přístup k médiím) v bezdrátových sítích je obtížný hlavně proto, že je téměř nemožné odeslat a přijímat data v jednu chvíli. Vysoká chybovost ještě více ztěžuje vytvoření dobře organizovaného komunikačního spojení na uzlech v bezdrátové síti.
* Efekty, které se podílí na plýtvání energií:
  + *Kolize –* zmařené úsilí, když dojde ke kolizi dvou přenášených paketů – detekce a reakce na kolizi, znovuodeslání dat, to vše vyžaduje další energii.
  + *„Přeslechnutí“ –* pokud uzel přijímá pakety, které jsou adresovány jinému příjemci ve stejném dosahu bezdrátového vysílání, potřebná energie pro přijmutí paketů (nebo aspoň jejich části) a detekce toho, že paket je ve skutečnosti adresován někomu jinému – to je vše zbytečné úsilí.
  + *Idle listening –* nečinné naslouchání; Protože přijímající uzel nemá vůbec tušení o tom, kdy začne odesílatel vysílat, musí proto příjemce být neustále ve střehu a čekat na možný příjem paketů (zde tím pádem plýtváme energií tím, že nic neděláme a jen čekáme)
  + *Režijní náklady*
* Návrh protokolů musí brát tyto faktory v potaz.

### Klasifikace MAC protokolů pro Wireless Radio Nets

* **Connection-based protkoly**
* **Conection-based protokoly s rezervačním mechanismem** – podpora přenosu v reálném čase za využití QoS záruk. Využívá předně mechanismus pro rezervaci šířky pásma. Protokoly jsou *synchronní* (vyžaduje synchronizaci času na všech uzlech v síti) a *asynchronní* (spoléhá se obvykle jen a relativní informaci o času). Např. MACA/PER (MACA s Piggy-Backed Reservation)
* **Connection**-**based protokoly s plánovacím mechanismem** – zaměření na plánování paketů na kanálu – pokud chce současně přistoupit ke kanálu více uzlů, nelze se vyhnout kolizím (řešení kolizí musí být podporováno protokoly)
* Nelze poskytnout záruky QoS (Quality of Service)
* Dva přístupy:
  + *Sender-initiated protokoly* – přenos paketů je iniciován odesílatelem
  + *Receiver*-*initiated protkoly –* příjemce iniciuje řešení rozporů na základě protokolu
* Je třeba počítat s dvěma základními problémy:
  + *Hidden terminal problem –* kolize paketů v důsledku souběžného přenosu uzlů, které nejsou v přímém dosahu odesílatele, ale jsou v přímém dosahu příjemce
  + *Exposed termial problem –* Neschopnost uzlu, který je kvůli přenosu blokován blízkým vysílajícím uzlem, přenášet data na další uzel



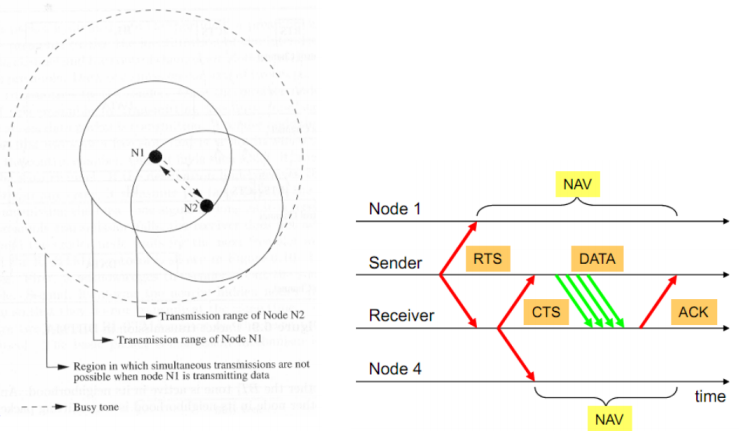
* Pro rezervaci kanálu máme dvě možnosi:
  + Příjemce informuje potenciální rušitele, zatímco probíhá proces přijímání
    - Tím, že odešle signál indikující rušení
    - Problém: Nelze využít stejný kanál, na kterém je aktuálně prováděno přijímaní
    - Např. Busy tone protokol
  + Příjemce informuje potenciálního rušitele předtím, než začne proces přijímání
    - Může být využit stejný kanál
    - Příjemce sám potřebuje být informován odesílatelem o tom, že dojde k přenosu
    - Potenciální rušitel musí dávat pozor na takovou informaci
    - Např. MACA protokol

**BTMA – Busy Tone Multiple Access**

* Přenosový kanál je rozdělen na datový a kontrolní kanál
* Když chce uzel odesílat pakety, požádá kanál, aby zkontroloval, jestli je busy tone aktivní. Pokud ne, zapne busy tone signál a začne přenášet data.
* Problém: velmi malé využití šířky pásma

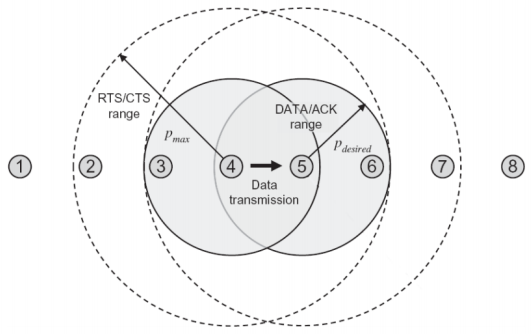
**MACA – Multiple Access Collision Avoidance**

* Dobře zavedené MAC protokoly v Ad-hoc doméně
* Využívá přídavné signalizační pakety: odesílatel se zeptá příjemce, jestli je schopný přijímat data – *Request to Send* *(RTS)*. Pokud příjemce může, odešle odesílateli *Clear to Send (CTS)*. Odesílatel odešle data, příjemce odešle příznak ACK.
* Potenciální rušitelé odposlouchávají RTS/CTS – RTC/CTS pakety nesou předpokládanou dobu odesílání dat
  + Rušitelé ukládají tyto informace v *Network Allocation Vector (NAV)*



**Power-Control MAC (PCM) protokol (pro informaci)**

* PCM poskytuje MAC vrstvě řešení pro kontrolu spotřeby energie tím, že mění energii přenosu dat za účelem redukce celkové spotřeby energie.
* **Idea:**
  + RTC/CTS handshake zprávy by měly být přenášeny s maximální dostupnou energií *pmax*
  + Handshake je využíván k určení potřebné energie k přenosu *pdesired*, která bude využita pro následující přenos příznaku DATA/ACK.
  + Výpočet *pdesired* je proveden od úrovně signálu přijatého RTC v kombinaci s nějakým známým minimálním prahem pro přijatou sílu signálu *Rxtresh*, který je nutná pro správné dešifrování zpráv.
* , kde *pr* označuje přijatou úroveň energie a konstantní parametr *c* je použit pro zvýšení *pdesired* podle podmínek okolího prostředí.
* Může být kombinováno s jakýmkoliv RTC/CTS založenným na MAC protokolu
* Přenos dat je ohraničen, takže neobtěžuje žádné další uzly.

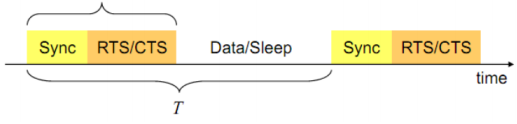


**Sensor-MAC (S-MAC) (pro informaci)**

* Nový protokol, který vylepšuje efektivitu energie v multi-hop rádiových sítích
* *Primární cíl:* udržet si flexibilitu connection-based protokolů, zatímco v multi-hop sítích se efektivita využití energie zlepší – idle listenning v MACA je částečně nevhodné v případě, že průměrná frekvence toku dat je nízká (většinu času se nic neděje)
* *Idea:* Vypnout uzly a ujistit se, že sousední uzly se současně zapnou, aby umožnily výměnu paketů („rendez-vous“)
  + Jen v těchto aktivních periodách může dojít k výměně paketů
  + K výměně je potřeba plán probouzení mezi sousedy
  + Když se uzel vzbudí, v podstatě provede RTS/CTS
  + Představuje hrubozrnný cyklus naslouchání a spaní s cyklem *duty-cycle D = , duty-cycle* = míra pro efektivitu energie uzlu



* Ve skutečnosti je perioda naslouchání rozdělena, aby byla podporována synchronizace mezi sousedy, stejně jako soutěž bezdrátového kanálu využívajícího RTC/CTS handshake mechanismus
* Všechny uzly si vybírají svůj vlastní naslouchací/spací plán – tyto plány jsou sdíleny s jejich sousedy, aby byla umožněna komunikace mezi všemi sousedícími uzly
* Každý uzel periodicky vysílá svůj plán v SYNC paketech, který poskytuje jednoduchou synchronizaci času
* Pro redukci režijních nákladů S-MAC podporuje sousední uzly v přijímání identických plánů
* Existuje mnoho dalších variant: T-MAC, B-MAC, P-MAC, Z-MAC atd.



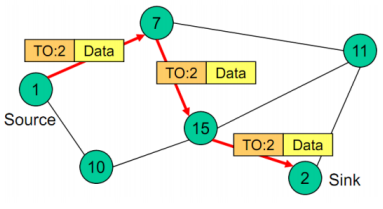
## Ad-hoc routování

* Typicky nejsou uzly v přímém komunikačním dosahu se všemi ostatními uzly
* Uzly potřebují objevit cesty (skládajících se z mezi-uzlů), přes které mohou doručovat pakety do cílové destinace = úkol routovacího protokolu
* Pohyblivost uzlů a chybějící centralizovaná kontrola tento úkol činí obtížnějším
* Routovací protokol musí být spárován s medium access control (MAC) protokolem – routing protokol specifikuje, komu by měl uzel posílat data. MAC protokol specifikuje kdy by se měl paket odeslat.

### Address-based routování vs. Data-centric forwarding (přeposílání)

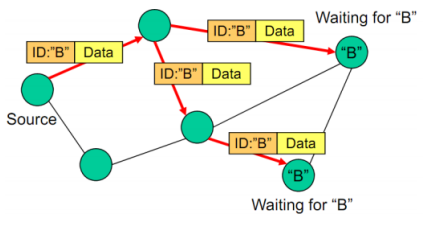
**Address-based routování:**

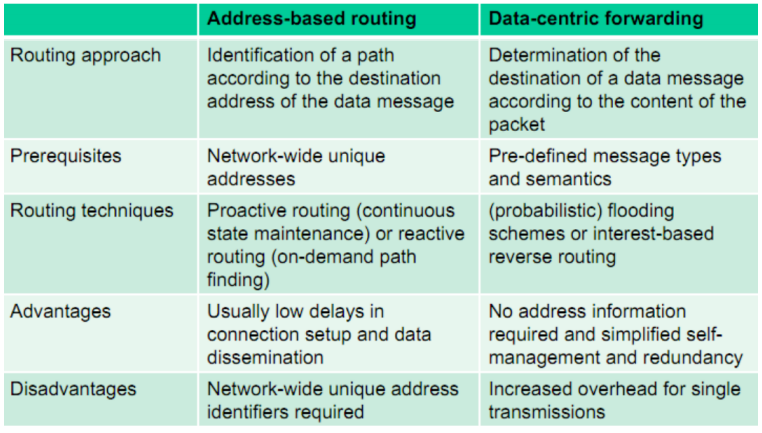
* První přístupy routování pro ad-hoc a senzorové sítě
* Zprávy jsou mířeny k dobře specifikované konkrétní destinaci (sink) – tyto routovací protokoly vyžadují, aby každý uzel měl unikátní adresu nebo identifikátor napříč celou sítí
* Poskytuje podporu pro unicast, multicast a broadcast zprávy
* Výhoda: možnost identifikace specifických uzlů a odeslání zprávy na tyto uzly



**Data-centric forwarding**

* Někdy (obzvláště v WSN) není aplikací vyžadována unikátní adresa, ale je vyžadována jen užitým routovacím protokolem.
* Adresovací schéma může být odebráno a nahrazeno specifickým významem (sémantikou) přenášených zpráv
* Adresy uzlů jsou vyměněny za druhy zájmů příslušných uzlů.
* Payload info (data, která se přenáší) jsou využita k přeposlání zprávy směrem k příslušné destinaci





### Address-based Ad-hoc Routování

Existuje mnoho různých klasifikací routovacích protokolů:

* **Proaktivní vs. Reaktivní protokoly:**
  + Proaktivní protokoly – Prozkoumávají cesty v síti ještě před tím, než jsou potřeba. Poskytují malou odezvu, ale velkou routovací zátěž.
  + Reaktivní protokoly – Prozkoumává cesty jen tehdy, když jsou potřeba. Vyšší latence (odezva), ale menší routovací zátěž.
* **Table-driven vs. Source-routing protokoly:**
  + Table-driven protokoly – každý uzel zná pouze další hop směrem k cíli. Routovací zátěž je malá, ale může dojít k routovacím smyčkám.
  + Source-routing protokoly – uzly znají celou cestu k cíli. Je jednoduché vyhnout se routovacím smyčkám, ale je zvýšená zátěž při routování.
* **Ploché vs. Hierarchické protokoly:**
  + Ploché (flat) protokoly – všechny uzly využívají stejné algoritmy (relativně stejné). Routovací zátěž může růst velmi rychle s tím, jak roste počet uzlů.
  + Hierarchické protokoly – některý uzly mají dodatečnou zodpovědnost (algoritmy jsou komplikovanější). Spolu s růstem sítě dochází k dobrému škálování výkonu.
* **Location-based vs. Non-location-based protokoly:**
  + Location-based protokoly – využívají fyzické lokace uzlů (redukují tak routovací zátěž). Uzly musí být vybaveny GPS nebo čímkoliv podobným.
  + Non-location-based protokoly – Fyzická lokace uzlů je neznámá. Routovací zátěž je typicky větší.

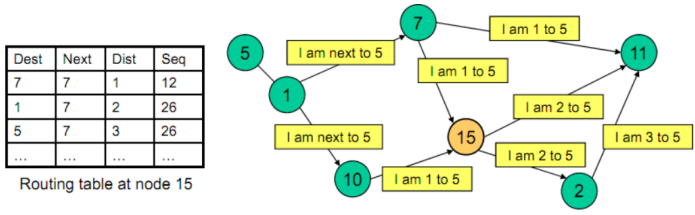
### Proactive Ad-hoc Routing

Proaktivní ad-hoc routovací protokoly:

* Spoléhají na periodické sbírání a výměnu informací o topologii
* Buď *table-driven* (distance-vector), nebo *link-state* mechanismy pro údržbu topologie
  + *Table-driven* (distance-vector) protokoly si periodicky vyměňují routovací tabulku se sousedními uzly.
  + *Link-state* distribuuje informace o topologii (updates), takže každý uzel si může optimální cestu v síti spočítat sám.
* Nevýhodou je velká routovací zátěž při údržbě stavu (topologie)
* Výhodou je, že datové pakety mohou být přeposílány kdykoliv a do jakékoliv destinace v síti.

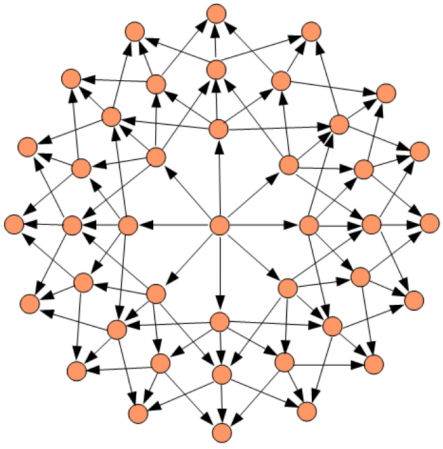
**Destination Sequence Distance Vector (DSDV)**

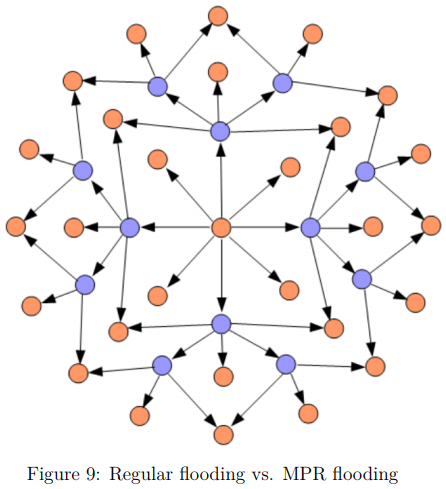
* Distance vector routovací protokol inspirován RIP protokolem
* Založeno na distribuované Bellman Ford funkci
* Uzly periodicky sdílí celé routovací tabulky mezi sousedy při změně topologie, je možné provádět inkrementální aktualizace cest v síti.
* Každý uzel ví „kde“ jsou všichni ostatní – routovací tabulka spravuje *O*(*N*) položek.
* Starší informace jsou použity pro udržování síťových cest čerstvých a pro předcházení routovacím smyčkám
* Jednoduchý protokol, ale není škálovatelný (vysoká zátěž)



**Optimized Link State Routing (OLSR) Protocol**

* Link-state-based protokol – informace o stavu spojení jsou rozšířeny do sítě => uzly si udržují informace o celé topologii a vypočítávají nejkratší cestu každý samostatně.
* Vysílání (rozšiřování informací o topologii) jsou optimalizována pomocí *MultiPoint Relays (MPRS)* – namísto pravidelného „zaplavování“ sítě, které představuje vysokou routovací zátěž, každý uzel si vybere a udržuje svůj vlastní MPR. Pravidlo: „Pro každé sousedy vzdálené 2 hopy musí existovat MPR, přes které se mohou navzájem kontaktovat.“
* MPR také dále povolují shromažďovat link-state informace.
* Výhoda: Vhodné pro velké a husté sítě
* Nevýhoda: nedostatečná bezpečnost, chybějící podpora pro multicast



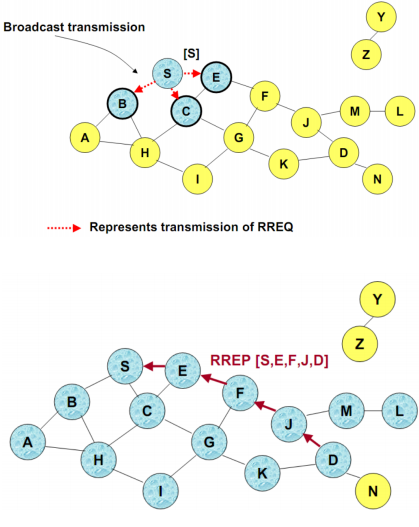


### Reactive (On-Demand) Ad-hoc Routing

* Jednotlivé síťové cesty jsou objevovány jen tehdy, když jsou potřebné (když je potřeba přenášet datové pakety)
  + Výhodou je úspora energie a šířky pásma během neaktivity.
  + Nevýhodou je další zpoždění, které vzniká při nastavování informací o routování
* Výhoda – nejsou potřeba (skoro) žádné routovací tabulky
* Nevýhody:
  + Velké zatížení sítě během zaplavování sítě během dotazování na síťové cesty
  + Se vzrůstající mobilitou klesá výkon

**Dynamic Source Routing (DSR)**

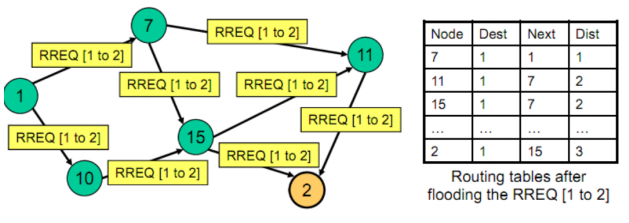
* Source-based (source = zdroj) routing protokol
* Využívá oddělené *Route Request (RREQ)* a *Route Reply (RREP)* pakety k objevování síťové cesty ze zdroje do cílové destinace
  + Odesílatel zaplaví síť pakety RREQ
  + Uzly přeposílají pakety RREQ, jakmile jim přidají svou identifikaci
  + Cílový uzel přijme RREQ paket a pomocí unicastu vyšle zpět k odesílateli RREP paket
* Routovací informace jsou uloženy v objevených paketech – není potřeba udržovat globálně validní routovací tabulky a informace o stavu každém uzlu
* Datové pakety jsou odeslány, jakmile byla vytvořena síťová cesta
* Dobře funkční pro menší a méně husté bezdrátové sítě.
* Optimalizace: cáchování síťových cest tím, že uzly vyhlíží RREQ a RREP pakety od ostatních uzlů



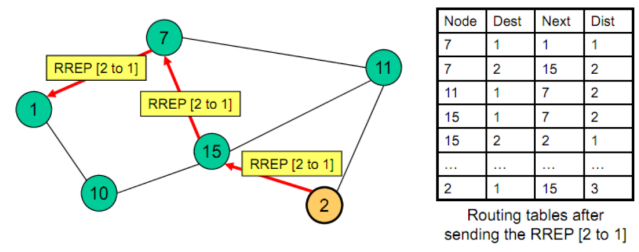
**Ad Hoc on Demand Distance Vector (AODV):**

* Další reaktivní routovací protokol. Vyhledává síťové cesty mezi zdrojem a cílem na požádání.
* V podstatě stejná procedura objevování síťové cesty je použita i u DSR, nicméně se musí vyrovnávat s přetížením paketů, které je zavedeno v DSR (source-routing – celá síťová cesta musí být uložena v každém paketu)
* V AODV si musí všechny uzly pamatovat odkud pakety pochází a tuto informaci ukládat do svých routovacích tabulek – tj. prostřední uzly v dané síťové cestě udržují routovací tabulky místo toho, aby využívali zdrojového routování. Když prostřední uzly znají síťovou cestu k cíli z předcházející komunikace, můžou odpovědět na paket RREQ namísto toho, aby to udělal cílový uzel.
* Výhodou je nízké zpoždění při nastavování připojení (v porovnání s DSR)

**Route Setup** – Síť je zaplavena RREQ pakety (omezeno TTL, které popisuje maximální síťový průměr)

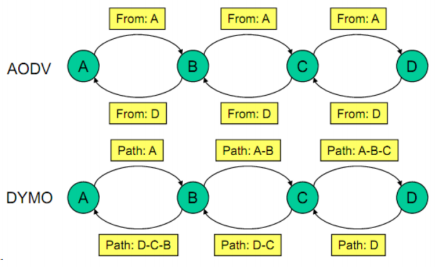


**Route Reply – Odpověď Síťové cesty** – The RREP pakety jsou pomocí unicastu poslány směrem ke zdroji



**Dynamic MANET On Demand (DYMO)**

* Následník AODV
* Redukuje přetížení sítě při nastavování síťové cesty a při údržbě síťové cesty



### Specializované routovací protokoly

**Geografické routování**

* Namísto udržování routovacích tabulek (k zjišťování pozice uzlů), odvozujeme pozici routeru z jeho fyzického umístění.
* Uzly znají svoje geografické souřadnice (GPS, lokační služba mapující ID uzlu na pozici uzlu atd.)
* Odešle zprávu sousedovi, který leží ve správném směru, jako další hop.
* Např. DREAM, GPSR, LAR

**Energy-Aware Routovací protokoly**

* Berou v úvahu kapacitu energie jednotlivých uzlů
* Mohou být využitý některé metriky za účelem minimalizace:
  + Spotřeby energie na každý paket (vybíráním síťové cesty s nejnižší energetickou náročností)
  + Odchylky od úrovní výkonu uzlů (vybíráním uzlů s vyšším výkonem)

# Počítačové sítě a multimédia

## Média, Multimediální aplikace

* *Multimédia* = informace/data, které jsou složena z několika různých typů/forem médií, které jsou nějakým způsobem integrovány dohromady.
* *Média* = text, obrázky, řeč/zvuk, video, interakce atd.
* *Multimediální aplikace:*
  + Pokročilé aplikace, které umožňují vzdálené sdílení zdrojů nebo interaktivní spolupráce
  + Příklady multimediálních aplikací: video telefonie/konference, multimediální elektronická pošta (obrázky, text, audio, rádio/televize (audio a video), real-time interaktivní prostředí pro spolupráci
  + Analogové audio a video signály by měly být převedeny do digitální formy
  + Za účelem redukování objemu informací, které májí být přeneseny, jsou použity patřičné kompresní algoritmy (ztrátová/bezeztrátová komprese)

### Charakteristiky medií

**Text:**

* Nejpopulárnější typ médií
* Šířeno mnoha formami – FTP, HTTP, SMTP
* Požadavky a šířku pásma závisí na jeho velikosti – můžou být dodatečně redukovány pomocí kompresních technik
* Charakteristika chyb záleží do značné míry na aplikaci – některé textové aplikace (např. přenos souborů) vyžadují, aby byla textová komunikace zcela beze ztrát a chyb => mělo by být použito TCP. Některé textové aplikace (např. instant messaging) můžou tolerovat nějaké chyby, stejně tak i ztráty => lze použít UDP
* Co se zpoždění týče, text-based aplikace obvykle nemají žádné požadavky na real-time přenosy (jako je např. hranice zpoždění nebo jitter).

**Audio:**

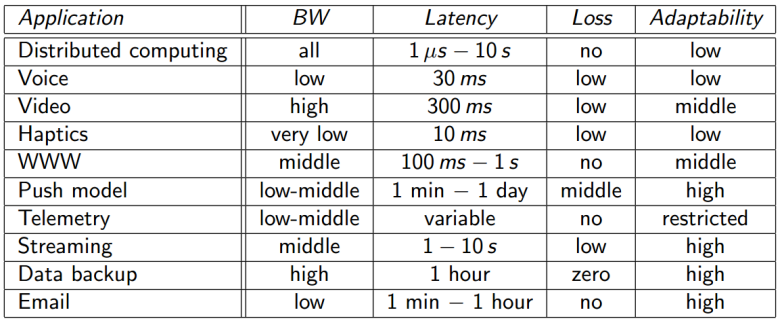
* Zvuk/řeč převedený do digitální formy využitím tzv. *samplingu* a *kvantování.*
  + *Sampling* – (=vzorkování) redukce plynulého signálu na diskrétní. Jednotlivé samply jsou skenovány v časových periodách (sample frekvence)
  + *Kvantování* – proces mapování/aproximace síly signálů jednotlivých samplů (vzorků) pomocí diskrétních symbolů nebo celočíselných hodnot
* *Požadavky na šířku pásma* digitalizovaného audio signálu závisí na dynamického rozsahu a/nebo na frekvenčním rozsahu – např. hlas v telefonu využívá redukci dynamického rozsahu (logaritmický A-law mechanismus)
* Audio média typ má volné požadavky na ztráty paketů a jejich chyby – může tolerovat více jak 2% ztrátu paketů, aniž by docházelo ke znatelné degradaci. Většina audio aplikaci navíc má zabudovaný mechanismus na vypořádání se se ztrátou paketu využitím interpolace.
* Požadavky na real-time/zpoždění striktně závisí na očekávané interaktivitě zúčastněných stran.
  + Např. Aplikace pro internetové telefonování/konference jsou vysoce interaktivní a vyžadují kratší čas odpovědi – zpoždění musí být přijatelné. *Real-Time Intolerant (RTI)* aplikace
  + Na druhé straně, např. živé internetové přenosy mají poměrně nízkou úroveň interkativity – vyžadují slabší omezení na maximální zpoždění. *Real-Time Tolerant (RTT)* aplikace

**Grafika a animace:**

* Sem patří statické (digitální obrázky) a dynamické typy médií (flashové prezentace)
* Nekomprimované a digitálně zakódované obrázky obsahují pole pixelů, , kde je každý pixel zakódován do určitého počtu bitů, které reprezentují jas a barvu. Obvykle větší, co se týče velikosti – proto se obvykle využívá komprese.
* Hodně moderních kompresních algoritmů pro obrázky jsou progresivní – to má důležité důsledky v případě přenosu dat přes komunikační síť.
* Když je takový obrázek přijat a dekomprimován, příjemce si ho může zobrazit v nízké kvalitě a poté vylepšit zobrazení následujících obrazových informací, které jsou přijaty a dekomprimovány později – tzv. *pyramid-coding* – obrázky jsou zakódovány do vrstev, kde první vrstvy mají nižší rozlišení a pozdější vrstvy (později přijaté) mají progresivně zvýšené rozlišení.
* Obvykle tolerantní k chybám (aplikace používané k renderování by se měli se ztrátou paketů vypořádat)
* Podobně jako u textových souborů nemají obrázky žádné požadavky na real-time přenos.

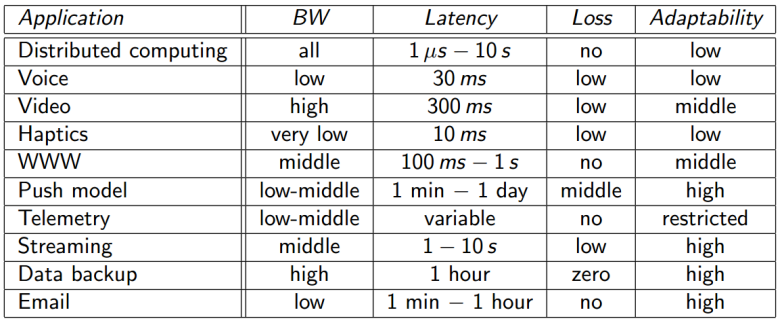
**Video:**

* Sekvence obrázků/rámců, které jsou zobrazeny v určité frekvenci (např. 24 nebo 30 FPS – frames per sec)
* Stejně jako u digitalizovaného audia je video přenášeno sítí jako proud diskrétních paketů.
* *Požadavky na šířku pásma* jsou obvykle velmi vysoké v závislosti na kompresi:
  + *Nekomprimované video –* obvykle větší velikost, ale umožňuje minimalizovat end-to-end odezvu
  + *Komprimované video –* menší velikost, ale kompresní metody zvyšují odezvu – proto toto není vhodné pro striktně real-time aplikace
* Požadavky na chyby a real-time charakteristikcy jsou podobné, jako v případě audia.



### Klasifikace médií

* Média mohou být buď **real-time** (RT) nebo **non real-time** (NRT)
  + RT média vyžadují buď pevné nebo pružné hranice end-to-end zpoždění/jitteru
  + NRT média (text, obrázky, …) nemají striktní omezení na zpoždění, ale mohou mít pevné požadavky na chyby.
* RT média mohou být dále klasifikována jako **diskrétní média** (DM) nebo **continuous media** (CM) – v závislosti na tom, jestli jsou data přenášena diskrétně (soubor nebo zpráva) nebo průběžně (proud zpráv se závislostmi mezi zprávami)
* RT-CM média můžou být dále klasifikována jako **delay tolerant** nebo **delay intolerant**, tedy tolerantní ke zpoždění nebo netolerantní ke zpoždění. V závislostí na tom, jestli média mohou tolerovat větší množství zpoždění bez znatelné degradace výkonu.



## Požadavky multimédii na komunikační síť

Můžou být obecně rozděleny na:

* **Požadavky na síťový provoz** – limity na real-time parametry (jako je zpoždění, jitter, šířka pásma a spolehlivost). Tyto požadavky mohou být splněny jen díky vylepšení základní Internetové architektury.
* **Funkční požadavky** – podpora multimediálních služeb (jako je multicasting, bezpečnost, mobilita a session management). Tyto požadavky mohou být splněny díky nově představeným protokolům na TCP/IP síťovém modelu.

## Požadavky na síťový provoz

### Požadavky na síťový provoz – Real-time charakteristiky (zpoždění + jitter)

* Real-time síťový provoz prosazuje striktní hranice na end-to-end zpoždění paketů a jitter.
* *Delay = zpoždění –* čas, po který paket cestuje ze zdroje do cíle.
* *Jitter =* nežádoucí odchylka, kolísání v rychlostech odezvy při odesílání paketů
* Výkon je vylepšen s tím, jak klesá zpoždění a jitter.
* End-to-end zpoždění je ovlivněno:
  + Zpožděním při zpracování paketů
  + Zpožděním při přenosu paketů
  + Zpoždění při propagaci
  + Zpoždění při routování a řazení paketů do fronty

**Zpoždění při zpracování paketů:**

* Konstantní zpoždění, ke kterému dochází jak u zdroje, tak v cíli – A/D, D/A konverze času a čas, který zabere paketizace skrz různé vrstvy protkolů
* Obvykle jedna z charakteristik operačního systému a multimediální aplikace – pro málo zatížený systém může být tato charakteristika zanedbatelná, nicméně se zvyšující se zátěži se stává i toto zpoždění znatelnější.
* Snížení toho zpoždění vyžaduje vylepšení softwaru zahrnující využití multimediálních operačních systému, které poskytují vylepšení technik správy procesů, souborů, zdrojů a paměti zároveň s real-time plánováním a vylepšováním aplikace.

**Zpoždění při přenosu paketů:**

* Čas, který si vezme fyzická vrstva na zdroji k tomu, aby přenesla pakety po lince.
* Závisí na více faktorech:
  + *Počet aktivních sezení –* fyzická vrstva zpracovává pakety v pořadí FIFO, což může být obzvláště znatelné tehdy, když operační systém nepodporuje real-time plánovací algoritmy podporující multimediální síťový provoz.
  + *Zpoždění přístupu k MAC –* zpoždění ovlivněno přistupováním odesílatele(ů) dat k médiu – vznikají kolize a ty vedou ke zpoždění služby. Široce rozšířené Ethernetové sítě nemůžou poskytnout žádné pevné záruky na toto zpoždění přístupu (kvůli nedeterminismům CSMA/CD přístupu)
  + *Přepínání kontextu v operačním systému –* odesílání nebo příjem paketu zahrnuje také přepínání kontextu v operačním systému – to trvá určitý konečný čas => existuje teoretický maximální čas, ve kterém může počítač přenést pakety. Redukce v tomto zpoždění vyžaduje vylepšení ovladačů zařízení a zvýšení rychlosti počítače.

**Zpoždění při propagaci**

* Čas přenosu paketů po přenosové lince – omezeno rychlostí přenášeného signálu (nejčastěji omezeno rychlostí světla)
* Kratší vzdálenosti znamenají kratší propagační časy
* Vzdálenost 200 m => propagační zpoždění 1 ms
* Vzdálenost 20000 km => zpoždění 100 ms
  + Fyzické limity, které nemohou být více redukovány (rychlost světla)
  + Interaktivní aplikace vyžadují, aby byl reakční čas menší než 100 – 200 ms
  + Pokud je propagační zpoždění větší, než tato hodnota, neexistuje žádné vylepšení, které by zvýšilo kvalitu interaktivního sezení.

**Zpoždění při routování a řazení paketů do fronty**

* Internet zachází s každým paketem stejně bez ohledu na to, jestli se jedná o real-time paket nebo non-real-time paket.
* Všechy mezilehlé cesty dělají nezávislé rozhodování při směrování každého příchozího paketu.
* Když pakety dorazí do fronty, musí čekat náhodně dlouhou dobu, než mohou být obslouženy – toto závisí na aktuální zátěži routeru.
* Zpoždění při routování a řazení paketů je náhodné, tím pádem v vzniká *jitter*
* Zpoždění může být omezeno službami Integrated Services (IntServ)/Differentiated Services (DiffServ), MPLS

### Požadavky na síťový provoz – šířka pásma

* Multimediální streamy mají obvykle vysoké požadavky na šířku pásma
* Komunikační síť musí být schopná zvládat takto vysoké požadavky na šířku pásma, aniž by při tom byla nespravedlivá k ostatním komunikačním tokům.
* Požadovaná šířka pásma může být redukována díky kompresním mechanismům, ale tyto mechanismy nemohou být použity pro všechny přenosy multimédií (např. interaktivní real-time video)
* Internet nenabízí žádný mechanismus, kterým by šlo rezervovat síťové zdroje splňující tak vysoké požadavky na šířku pásma – to je ponecháno na uvážení aplikace, aby se dynamicky přizpůsobila aktuálnímu zahlcení sítě.
* TCP-based aplikace využívají výhody vestavěné kontroly zahlcení. Multimediální aplikace jsou však většinou založeny na UDP, které nemá žádný mechanismus na kontrolu zahlcení
* Je nemožné odstranit tyto nedostatky v dnešním Internetu
* Vylepšený model Internetových služeb vyžaduje:
  + *Admission control = kontrola přístupu:* Aplikace musí nejprve získat oprávnění k odeslání dat danou frekvencí na základě daných charakteristik síťové dopravy
  + *Rezervace šířky pásma:* Pokud je přístup udělen, příslušné zdroje (vyrovnávací paměti, šířka pásma) budou po síťové cestě rezervovány
  + *Mechanismus kontroly provozu:* K zajištění toho, aby aplikace neodesílala data rychleji, než bylo vyjednáno.
* Cena operací protokolů pro zpracovávání se více týká rychlosti zpracování paketů, než šířky pásma
* Zpracování paketů je hodně závislé na počtu paketů a méně závislé na velikosti paketů => rychlost zpracování paketů je důležitá metrika.

### Požadavky na síťový provoz – Požadavky na chyby

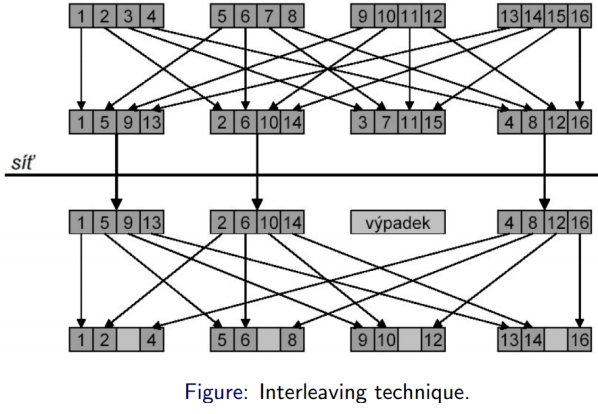
* Různé typy médií mají zcela odlišné požadavky na chyby (chyba = ztracený nepoškozený paket) – od situace, kdy je médium zcela netolerantní vůči chybám až po situaci, kdy je médium poněkud tolerantnější. Internet nemůže nabídnout záruky na charakteristiky chyb, protože cesta, kterou paket následuje, není fixní => odesílatel nemá žádnou znalost o chybových charakteristikách sítě.
* Nejčastější metody pro opravu chyb:
  + Sender-based oprava – **aktivní oprava** – Automatic Retransmissio Requst (ARQ) mechanismus, vhodné pro error intolerant aplikace. **Pasivní oprava** – Forward Error Correction (FEC), Interleaving
  + Receiver-based oprava – Error Concealmet

### Požadavky na síťový provoz – Forward Error Correction (FEC)

* Vyžaduje, aby k paketům byly přidány extra informace – tyto informace jsou použity pro obnovu ztracených paketů.
* **Media-independent FEC** – využívá blokové nebo algebraické kódování k produkci dodatečných paketů, které pomáhají při obnově ztrát.
  + Kódování berou kódové slovo *n* datových paketů a generují tak *m* dodatečných kontrolních paketů
  + *Parity Coding –* operace XOR je aplikována na skupinu paketů za účelem generování paritních paketů
  + *Reed-Solomon Coding* – polynomy s určitými číselnými základy
  + Nevýhoda – způsobuje dodatečné zpoždění a zvyšuje využití šířky pásma
* **Media-dependent FEC** – využívá charakteristiky médií
  + Každá datová jednotka je poslána ve více paketech – primární šifrování + sekundární šifrování (může být nižší kvality než primární šifrování)
  + Nemusí vyžadovat přenos FEC pro každý paket, díky povaze média
  + Výhoda – nízká odezva => vhodné pro interaktivní aplikace
* Pro aplikace je výhodné, aby znaly charakteristiky chyb komunikační sítě => může být využit adekvátní úroveň FEC, např. bezdrátové vs. Drátové sítě

### Požadavky na síťový provoz – Interleaving (=prokládání)

* Může být využito jen tehdy, když velikost jednotek v médiu je menší, než je velikost paketu (např. audio jednotky). End-to-end zpoždění není důležité.
* Na základě znovuobnovení jednotek v médiu před síťovým přenosem (původně přilehlých jednotek oddělených garantovanou vzdáleností) a navrácení do původního pořadí u příjemce – ropztýlení efektu ztráty paketů
* Ztráta jednoho paketu způsobí mnoho malých mezer v původním médiu (jednodušší k vyřešení)
* Výhoda – nezvyšuje využití šířky pásma
* Nevýhoda – zvyšuje se odezva => nevhodné pro interaktivní aplikace



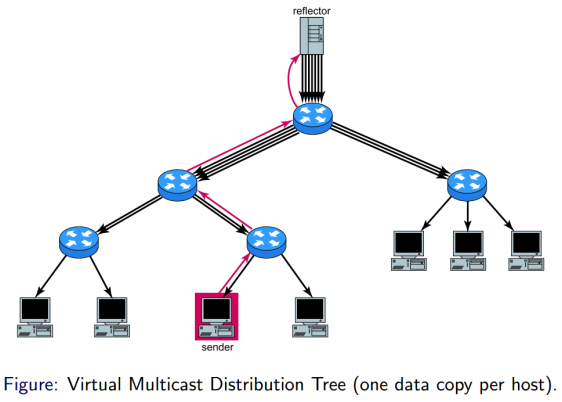
### Požadavky na síťový přenos – utajení chyb

* Předvídání/kompenzace ztracené informace na základě správně přijatých paketů
* Využíváno mnoha error-tolerant aplikacemi
* Funguje dobře pro relativně malou míru ztrát ( < 15 %) a pro malé pakety (4 – 40 ms)
* Několik typů (se vzestupným pořadím výpočetní ceny a vylepšeného výkonu):
  + *Insertion-based* – vkládání předvyplněných paketů, které obsahují ticho, hluk ebo opakování přilehlých paketů
  + *Interpolation-based –* Vyhledávání vzorů a hledání přibližných hodnot za účelem odvození chybějících paketů
  + *Regeneration-based* – odvození stavu dekódování z paketů obklopujících ztracený paket a generování ztraceného paketu na základě takto získaných informací

## Funkční požadavky

### Funkční požadavky – podpora multicastingu

* *Multicast =* doručení zpráv skupině cílových počítačů v jeden čas během jednoho přenosu ze zdroje, dále automatické vytvoření kopií ve vnitřních elementech sítě (např. routery) jen tehdy, když to vyžaduje topologie sítě.
* Majetek sítě (hop-by-hop, ne end-to-end služby)
* Nicméně nedostupné v mnoha sítích
* Využívané protokoly: IGMP, DVMRP, MOSPF, PIM atd.
* => Může být kompenzováno multicastem aplikační vrstvy – jeden (nebo více) speciálně zaměřených uzlů/aplikací v síti zrcadlí příchozí síťový provoz. Poskytuje dodatečnou funkcionalitu (zpracování dat, kontrola přístupu, moderování atd.)



### Funkční požadavky – Bezpečnost a mobilita

**Bezpečnost:**

* Poskytuje následující pohledy na multimediální data:
  + *Integrita* – data nemohou být změněna během cesty k příjemci
  + *Autenticita* – data přichází ze správného zdroje
  + *Šifrování* – Data nemohou být dešifrována třetí stranou
* Zajištěno IPSec/IPv6 – **Authentication Header**  (AH)a **Encapsulated Security Payload** (ESP) protokoly

**Mobilita** – skoro stejné, jako mobilita u IPv6

### Funkční požadavky – Session management

Funkcionalita Session managementu zahrnuje:

* *Popis médií:*
  + Umožňuje distribuovaným multimediálním aplikacím distribuovat informace o session – např. typ média (audio, video nebo data) využitá v rámci session, schémata šifrování médií (PCM, MPEG II), čas session start/stop, IP adresa zúčastněných hostů atd.
  + Obvykle je nezbytné popsat celou session ještě před jejím založením
  + Např. Session Description Protocol (SDP), Session Initiation Protocol (SIP)
* Oznámení o session:
  + Umožňuje účastníkům oznamovat budoucí sessions – např. oznamování plánovaných pořadů na internetových rádiových stanicích
  + Např. Sessions Announcement Protocol (SAP)
* *Identifikace Session:*
  + Multimediální session se často skládá z více mediálních streamů (audio, video, text, obrázky atd.), které musí být identifikovány separátně – např. odesílání audia a videa jako dvou oddělených streamů, které musí příjemce dekódovat synchronně rozdělením kvality A/V streamu na nízkou a vysokou šířku pásma
  + Např. Real-Time Transport Protocol (RTP)
* *Kontrola Session:*
  + Protože můžou multimediální sessions zahrnovat více mediálních streamů, jejichž informace jsou často vnitřně propojené, multimediální komunikační sít musí zaručovat údržbu takových vztahů u koncového uživatele = *Multimedia Synchronization*
  + Navíc mnoho Internetových multimediálních aplikací může chtít kontrolovat přehrávání kontinuálního média např. zastavením, pozastavením, spuštěním přehrávání atd.
  + Např. Real-Time Control Protocol (RTCP), Real-Time Streaming Protocol (RTSP)

## Session management Protokoly

### Session Description Protocol (SDP)

* Autorem je IETF (Internet Engineering Task Force)
* Jedná se formát popisující parametry streamovacích média (typ média, šifrování média, …)
* Má více popisnou syntaxi v textovém formátu, než protokol
* SDP zprávy obsahují sérii linek (nazvaných pole)
* Každé pole má formát <tag>=<value>, tag – předdefinovaná jednopísmenná zkratka

### Session Announcement Protocol (SAP)

* Používáno pro oznamování multicast konferencí a jiných multicast sessions
* Využívá SDP k popisu session(s)
* SAP hlasatel periodicky pomocí multicastu posílá pakety na dobře známé multicastové adresy a porty – se stejným rozsahem jako je oznámeno session => příjemci tohoto oznámení jsou také potenciální příjemci oznamovaného session.
* Obsahuje mechanismus pro: zajištění integrity oznamování sessions, autentifikace původce oznámení, šifrování takových oznámení

### Real-Time Transport Protocol (RTP)

* Běží na vrcholu UDP (obvykle implementováno v rámci aplikace)
* Přenáší části (audio/video) dat a poskytuje:
  + *Sequencing –* pořadové číslo v RTP hlavičce pomáhá detekovat ztracené pakety
  + Identifikace payloadu – popisuje šifrování média
  + *Frame indication* – video a audio je posláno v logických jednotkách nazvaných rámce (frames)
  + *Source identification* – aby šlo identifikovat původce rámce v multicast session, je k dispozici identifikátor Synchronization Source (SSRC)
  + *Intramedia synchronization* – za účelem kompenzace různých zpoždění a jitteru pro kaety v rámci jednoho streamu jsou k dispozici časová razítka (označení času pro pakety)
  + Dodatečné informace o médiích můžou být vloženy využitím profilových hlaviček a rozšíření.
* RTP nemůže zajistit kvalitu přenosů, ale poskytuje aplikacím zdroje, které to umožňují

### Real-Time Control Protocol (RTCP)

* Kontrolní protokol, který pracuje ve spojení s RTP
* Sbírá statistiky o aspektech kvality distribuování médií během session a přenáší tato data do zdrojového média a k ostatním účastníkům
* Účastníci pravidelně odesílají RTCP pakety, aby získali zpětnou vazbu o kvalitě přijímaných dat.
* Poskytuje statistiky nad rámec standartních statistik a také kontrolní informate pro RTP tok dat – pakety odeslány, ztraceny, jitter, round-trop čas atd.
* Zdroje můžou tento protokol využít pro úpravu svých datových frekvencí a pro detekci chyb přenosu. Přiložené časové informace mohou být využity pro synchronizaci
* Ostatní zahrnuté informace – emailová adresa, telefonní číslo, jméno atd. Umožňuje uživatelům znát identitu ostatních uživatelů v rámci session.

### Real-Time Streaming Protocol (RTSP)

* „Nadrámcový“ kontrolní protokol, který umožňuje přehrávači médií kontrolovat přenos média streamu (pause, resume atp.)
* Přenos streamovaných dat samotných není úkolem RTSP protokolu
* Pracuje s RTP (a RSVP) a tím poskytuje kompletní streamovací služby do Internetu
* Podobné jako http v některých hlediscích, s tím rozdílem, že http je bezestavové, zatímco RTSP je stavové (identifikátor je použit, když je potřeba sledovat konkurenční session)

Konec 😊