# PA159 – Počítačové sítě

Jogi, mé zápisky z přednášek podzim 2013, obrázky z prezentací paní Hladké
*(v případě závažné chyby (nikdo není neomylný) nebo pro poděkování :) - twitter jogi1j nebo 100200@seznam.cz, díky)*

**[1] Počítačová síť –** skupina počítačů komunikující mezi sebou, velká část zaujímá režie poč.
také výměna (text, speech, video), sdílení HW, souborů, dat, inf, SW

- negativa – zneužití, skype apod.
- základní charakteristiky sítě – doručení, správnost doručení, včasné dodání

Ideál:

- pouze end-to-end
- nelimitovaný objem dat
- bezeztrátový přenos
- bez zpoždění
- zachování pořadí
- neporušitelnost dat

Reálná síť:

- vnitřní struktura
- omezená kapacita
- data se ztrácí
- občasné zpoždění
- nezachovává pořadí
- data se občas poruší

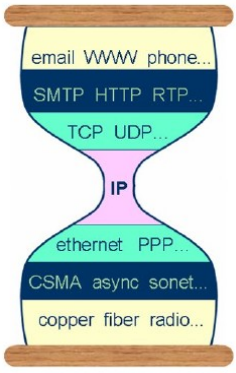
Požadujeme: efektivita, férovost, decentralismus, přizpůsobitelnost, multiplexing/dem., učenlivá (poučení z chyb)

Sítě spojované (vytváříme spojení mezi dvěma body) – nejdříve navázáno spojení, potom se přenášejí data, spojení mohou být permanentní (nemusí), efektivita ! – není dobrá, až nevhodná. Příklad – dříve telefonní systém, každý hovor spojen. *Jitter – kolísání velikosti zpoždění paketů při průchodu sítí (vzniká např. na směrovačích (routerech) jako důsledek změn routování, chování interních front routeru atd.)*
Síť nespojované – data nejdou přes jedno spojení, můžou být fragmentovány – rozděleny na měnčí. Neznáme stav sítě! Nevíme jak vypadá či jak vypadala. Implementace kvality služby nelze. Pracují ale efektivně.

Komunikační protokoly:
- tzv. virtuální spojení – nejprve requset, až odpoví response, můžu posílat
- protokol se dodržuje i v řeči ( Ahoj, ahoj,... )
- př. UDP, TCP, IP, IPv6, SSL, TLS, HTTP, FTP, SSH, Aloha, CSMA/CD...
- syntax, sémantika a časování
- síťový protokol je množina pravidel, které definují formát a pořadí zpráv, které si vyměňují dva či více komunikačních systémů

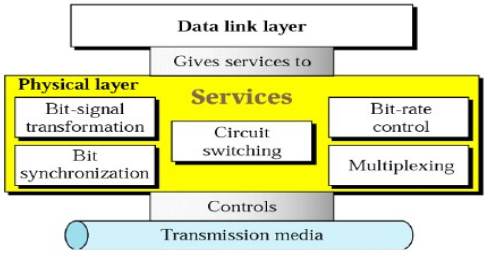
*entita – někdo kdo posílá nebo přijímá data*
*RFC – request for commnets – doporučení (ne standarty) pro internetové sítě či protokoly*

Standardizace – norma s cíle
kvalita, bezpečnost, kompatibilita, vzájemná spolupráce, přenositelnost
Dvě kategorie – de facto (dobrovolně – je to výhodné) x de jure (legislativa)
př. ISO, ITU-T, ANSI, IEEE, IETF (RFCs), IEC, etc.

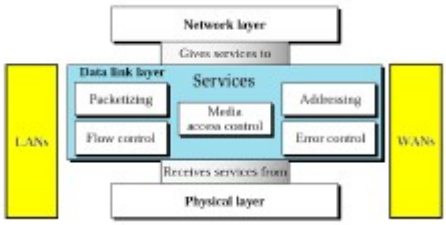



ISO/OSI – 7 vrstev, organizace OSI, aby se vědělo, kdo je za co zodpovědný , snadnější práce, teoretický model
~ aplikační, prezentační, relační, transportní, síťová, datová (linková), fyzická
TCP/IP model – 4 vrstvy, ~ aplikační, transportní, síťová, přístupu v síti

→ Fyzická vrstva – médium, posílání a přijímání dat (signálů), datové kódování do signálů , vrstva má za úkol tyto signály přenášet, použití multiplexingu, hlavní úkol přenášení bitů mezi dvěma uzly. Mnoho standardů (pro optiku, pro dráty apod.).
~ Bit-to-Signal Transformation
~ Bit-Rate Control
~ Bit Synchronization
~ Multiplexing
~ Circuit Switching
Transformace signálů analog/digitál. Drát/Bezdrát (metalické/optické), dnes optika ne sklo, ale umělá hmota. Drát < 10Gbit, kroucená dvojlinka.
Multiplexing & Demultiplexing
Analog – Frequency-Division Multiplexing (FDM) & Wave-Division Multiplexing (WDM)
Digitální: Time-Division Multiplexing (TDM)



→ Vrstva datového spoje (linková) – obdrží pakety a přenese na rámce po sdíleném médiu, umožňuje propojit nejen dva body, ale propojit celou lokální síť sdíleným médiem.
~ rámce – základní datová jednotka L2




~ adresace – MAC adresy, dodává výrobce
Jiné zákony pro drátové / jiné pro bezdrátové. Abychom zabránili kolizím.
Protokoly:
random-access protocols – Aloha, CSMA/CD, CSMA/CA
controlled-access protocols – based on reservations, polling, tokens, etc.
kanálové protokoly (multiplex-oriented access) – FDMA, TDMA,etc.
Propojení do topologie: bus, circle, star, tree, mesh, etc. Používáme: bridge (mosty) a switche (přepínače).
Backward Learning Algorithm – the bridge "learns" the locations of network stations (nodes) by listening on the media (observing the source addresses). Lze vytvářet cykly, používáme Distributed Spanning Tree Algorithm. Pokud most dostane od souseda zprávy, vyhodnotí nejkratší cestu (nemusí být fyzicky nejkratší / hledá se nejmenší cena / preference podle adresy), vybereme kořen a budujeme pomalu strom. Jak vybrat kořenový můstek – všichni prohlásí. Že jsou kořenový, kořenový s nejnižší adresou je zvolen a ostatní přestanou být kořeny. Strom pak postupně roste.

~ zprávy chyb – redundance dat (vysíláme více dat)
ARQ – požádání o další rámec v případě chyb
CRC – L2 a L4 prochází více vrstvami
FEC – detekce na bázi Hamiltonova kódu
~ detekce zahlcení

→ Síťová vrstva – servis pro transportní vrstvu, obdrží segmenty a přenese je do paketů, rozsáhlé sítě WAN, propojení LAN do větší sítí, nazývaných už internetem.
~ Internetworking – máme různé fyzické sítě, které propojujeme do internetu. Vrstva poskytuje iluzi jedné sítě.
~ Packetizing – obdrží segmenty a přenese je do paketů
~ Fragmentace – MTU maximální velikost dílů/ fragmenty. Pokud síť je s menší propustností, pak fragmentace na 2díly.

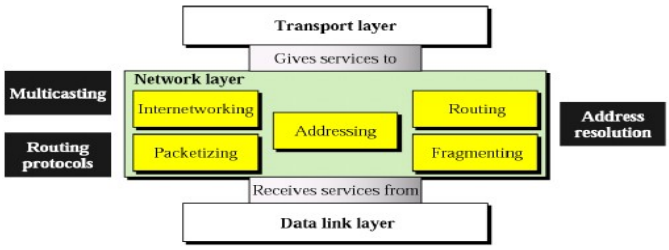
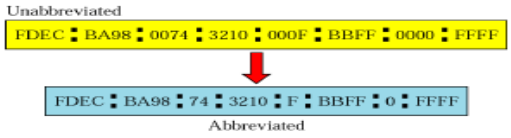
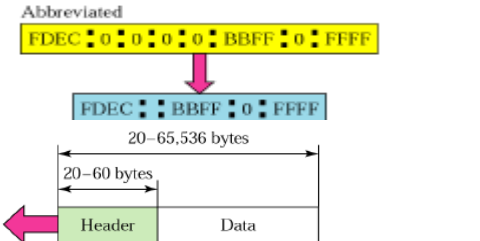


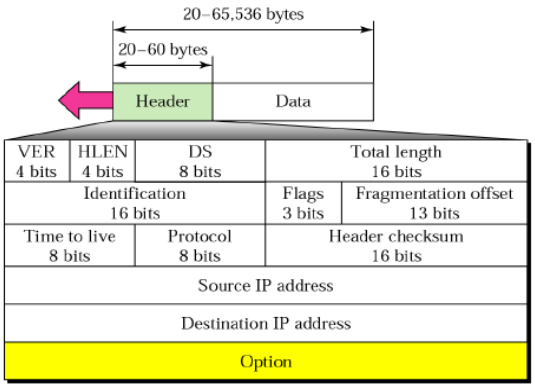
Figure: Position of the Network Layer.

~ Adresace – IP adresa unikátní, IPv4 (32b) – Unicast, Broadcast, Multicast (až na síťové vstvě přeložena na konkrétní adresy)
IPv6 (128b = 16Bites = 32 Hexadecimálních) – dostatek adres, pravidla:




← vypuštení nul
→ vypuštění :: lze jen jednou !

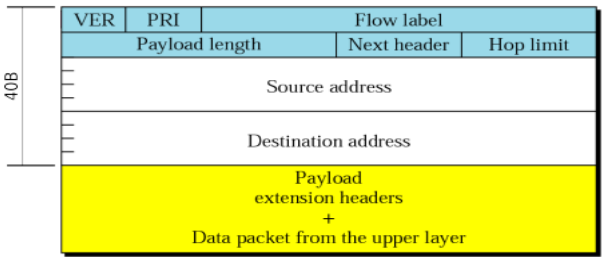



Unicast, Muticast (prefix ff00::/8), Anycast [novinka není v IPv4] – místo broadcastu.
IP protokol – nejrozšířenější, data transportována podle adresy, nespojovaná komunikace, používá – ICMP, ARP, RARP, IGMP. Standardizovaný protokol IPv4 ('81), IPv6 ('98)

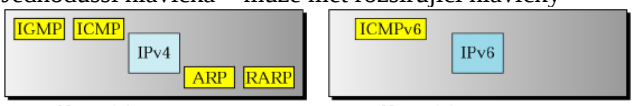
IPv4 – Version (v4, v6), Hlavička, Differentiated services (DS) – QoS, délka celkem, identifikace, Flags = Offset, TTL – kolikrát může paket projít (když vznikne cyklus → eliminace), Protokol ( 1 – ICMP, 1 – IGMP, 6 – TCP, 17 – UDP), Header checksum – jednoduší než počítat celý rámec, je lepší přepočítávat jen hlavičku, IP source and destination = 32bit IP, Option – testing a debugging, data.

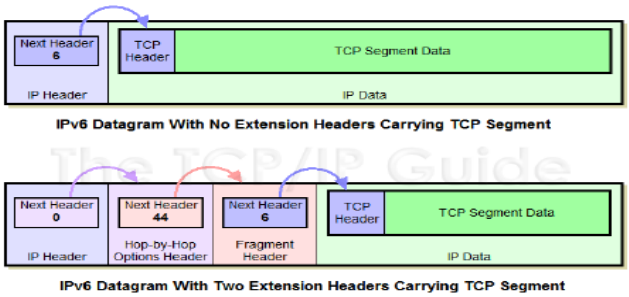
IPv6 – snaží se poučit, jednodušší hlavička, rychlejší zpracování, podpora přenosu v reálném čase, priority toku, bezpečnost, autokonfigurace (mobilní zařízení), a další.



Fixní hlavička 40B, není tu checksun, option ani další fragmenty = IPv6 nefragmentuje uvnitř sítě. Složení – version, priorita, datový tok (moc se nepoužívá), velikost, další hlavička, Hop limit ~= TTL, zdrojová – cílová adresa,

Jednodušší hlavička = může mět rozšiřující hlavičky




~ Směrování – najít cestu mezi dvěma komunikujícími uzly. Jedinou informací je dvojice adres (z kama kam). Snažíme se aby byla cesta optimální (topologie – statický faktor / zátěž – dynamický faktor).

Na síť se díváme jako na menší části – směrování teda probíhá krok za krokem s lokálními informacemi. Z globálního pohledu nemusí být optimální. Princip hop-by-hop. Optimální kritérium jsou metriky. Dva způsoby – statické (neadaptivní tabulky, malé systémy) x dynamické (složité distribuované algoritmy; centralizované/izolované/distribuované). Směrování je problém grafové teorie.

Nejčastější přístup směrování je:
Distance vector (DV) – vymění kompletní kopie směrovacích tabulek (Bellman-Ford)
Link State (LS) – periodicky si vyměňují seznam linek se kterými jsou spojedy (Dijkstra)

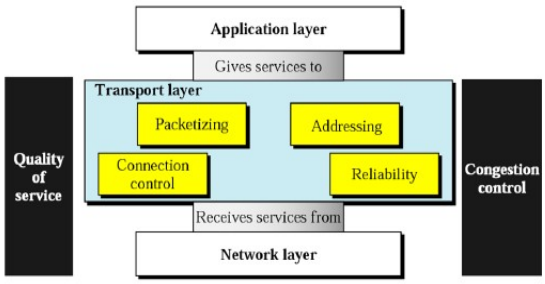
[DV] RIP protokol – identifikace pomocí CIDR, počet hopů(přechodů) použit jako metrika, nekonečno = 16 (omezení), peoriodická výměna, timeout 180s,
[LS] Open Shortest Path First (OSPF) – nejpoužívanější LS protokol, metrika cena, rozšiřován (bezpečnost, balancování více linek apod.), až 65tisíc metrika

~ Address Resolution – ARP, RARP (překládací protokoly mezi MAC a IP adresou)
~ Control Messaging – ICMP protocol

Autonomní systémy, kde se nahlíží na uzly jako na jeden. Máme také administrativní subjekty – CESNET, PASNET nebo i MU s podsystémy například FI a pak lokální rozdělení. Autonomní systémy dělíma na: *Stub (připojedny jen na jednom místě) x *Multihome (více vstupních míst, vysílá data které pouze vyprodukoval) x *Transit (více míst, přenos může být přes ně přenášen). Border Gateway Protocol, Path Vector (výměna popisu cesty – pomocí TCP, pro agregaci cest CIDR), velkou roli hrají politiky, směrovací pravidla. CIDR – je aktuální adresní schéma a mechanismus pro přidělování adres sítě Internet. Před zavedením CIDR byly adresy rozděleny do tříd (viz IP adresa) a koncovým sítím připojeným k Internetu se v závislosti na jejich velikosti přidělovala adresa sítě třídy A, B nebo C. CIDR přinesl do adresace dva nové principy řešící výše popsané problémy:délka adresy sítě je libovolnáadresy se přidělují hierarchicky, což umožňuje agregaci směrování.
Multicast – paralelní komunikační schéma, mám zdroj odkud putují data v jedné kopii. Putují sítí a jak je rozdělení tak se rozdělí, ale jinak se neduplikují. Nezajištěná služba. Má své vymezené adresy stejně tak v IPv6.

Rozdělení multicastu:
Source Based Tree
- první zavedený
- shoda dolů
- periodicky
- rušení uzlů bez členů
- TTL
- mínusy: záplavy, režije
- nepoužívá se
- DVMRP (RIP), MOSPF (OSFP), PIM-DM

Core Based Tree
- dnes používaný
- zdola nahoru
- klient kontaktuje MP
- redukce vysílání – větší stabilita, lepší škálovatelnost
- mínusy: závislost na jádře
- CBT, PIM-SM

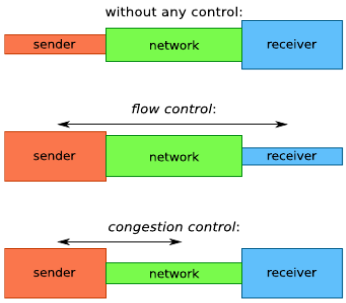



→ Transportní vrstva – poskytuje služby aplikační vstvě, odebírá data a dává je do segmentů a poskytuje je níže. Nebo taky přijímá pakety a poskytuje je v podobě segmentů aplikacím. Cílem je přenést data od jedné aplikace ke druhé.

Propojujeme dva koncové body.
~ Paketizace – data poskytována aplikacím jsou ve formě paketů.
~ Řídící kontrola – spojovaná či nespojovaná služby
~ Adresace – adresujeme jednotlivé služby/aplikace, přidáváme číslo portu, které definuje službu na jednotlivých bodech.
Spojovaná služby – provede se spojení, data jsou číslovány, služba umožňuje opravnými mechanismy data znovu vyžádat, aplikace má teda kopii bez drát.
Nespojovaná služby – může dojít ke ztrátám, může dojít ke ztrátě paketu, pakety se nečíslují.

UDP – User Datagram Protocol – poskytuje nespojovanou, nezajištěnou vtstvu. Spojení aplikace – aplikace, jednoduchý protokol pro chyby. Pokud aplikace vyžaduje všechna data, musí si to zajistit jinak sama. Je jednoduchý a má minimální režiji, nezajišťujete spojení, není třeba komunikovat a proto transportní hlavička je krátká a jednoduchá. Nevýhoda je, že UDP se nedá použít vždycky. Header(zdrojový a cílový port 16bx2, celková délka 16b, checksum 16b) + data

TCP – Transmission Control Protocol – Plně zajištěná a spolehlivá služba. Proběhnout 3 zprávy na který je zajištěno spojení a teprve potom se začínají převádět data. Piggybacking – kontrolní data. Dvoubodová propojení, není podporován multicast. Multiplexing/demultiplexing shodný s UDP. Header (zdroj. a cíl. port 16bx2, sekvenční číslo 32b, číslo potvrzení – očekávaného paketu 32b, délka hlavičky 4b, rezervace 6b, 6b řídících informací (obr), velikost okna 16b, kontrolní součet – checksum 16b, urgentní data 16, options – příznaky) + data.
Zahlcení můžeme mít dvojího typu – zahlcení odesílatele a zahlcení příjemce.



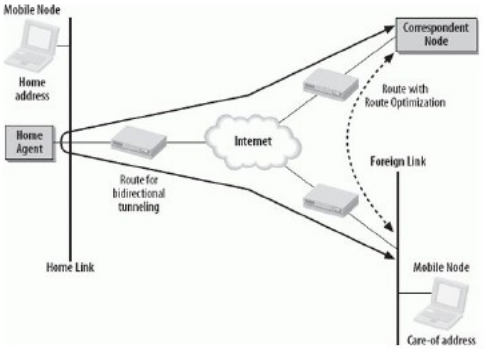

~ Spolehlivost
~ Kontrola zahlcení a QoS (Kvalita služby)

→ Aplikační vrstva – Důvod tvorby sítě. Komunikace – peer-to-peer nebo klient-server (centralizovaná). Aplikace s malými požadavky na síť a aplikace s velkými požadavky na síť. Protokoly: jmená služba (DNS) World-Wide-Web (HTTP) electronic email (SMTP) file transfer (FTP) multimedia transmissions (RTP/RTCP).

**[2] IPv6** – došli IPv4, i přes CIDR a NATy, vyčerpali se, svět běží dál a čeká se na přestup na IPv6. Ačkoliv IPv6 ještě není mnoho používána, tak už teď zaostává za vývojem sítě a dala by se vylepšit. IPv4 nemá zabudovanou bezpečnost. Není to jen více adres, ale také pokus o pokročilejší architekturu. 128bitová adresa. IPv6 má základní hlavičku a další rozšiřující se dají přidávat. Základní hlavička je jednoduchá a rychlá. Rozšíření – např. podpora real-time. Vychází se, že zařízení je někdy „doma“, kde je domovská síť, domácí agent. Podpora stavové/nestavové autokonfiguraci.
Neobsahuje (IPv4 obsahuje) – kontrolní součet (hop nelze), fragmentace (dochází na zdrojovém uzlu ne na hlavičce – ulehčení práce směrovači). Hop-by-hop potřebují směrovače, proto je uvedena hned za hlavní hlavičkou, fragmentace je až na konci jako poslední. IPv6 má 128bitové adresy. Zápis hexadecimální. Pravidla viz předchozí text IP. Hierarchie IPv6 je rozdělena na tři části. 64Bitů interface, 64bitů subnet (16bitů 2001, 16bitů RIR, 16bitů LIR, 16bitů subnet), nbitů globánlní prefix. Adresace je beztřídní > CIDR. Typy adres – unicast, multicast (data doručovány všem členům skupiny z jedné ip – ff00::/8) a anycast (data pouze pro nejbližšího souseda), chybí broadcast = v IPv6 multicast.
Fragmentace MTU – ICMPv6's Packet too big message.

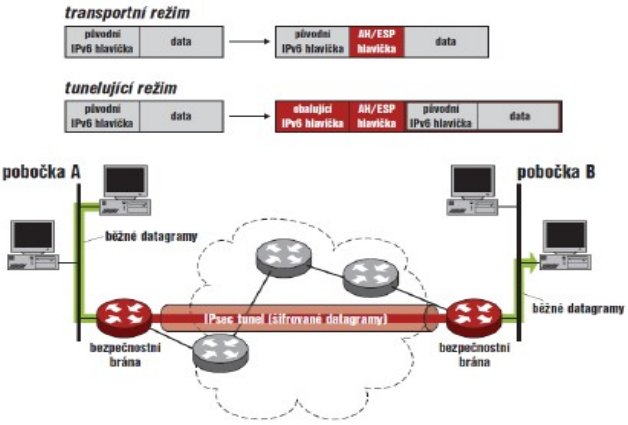
Neighbor Discovery Protocol (NDP) – (objevování souseda – místo ARP v IPv6) Použití pro L2 adresy. Odhalení duplikátních adres (důvěřuj ale prověřuj) a navíc zjišťujeme dostupné sousedy, autokonfigurace IP adres, jak stavovou tak bezstavovou. Základem tohoto protokolu je 5 ICMP zpráv: výzva směrovači, odpověď směrovače, výzva souseda, výzva souseda a ICMP přesměrování. Rozešle se zpráva obsahující IPv6 a pokud se IP adresa schoduje, tak se ozve a pošle linkovou adresu, rozesílání multicastem jen pro určitou skupinu.
~ Duplicate Address Detection (DAD) – detekce duplikátu adresy, porovnání linkové a pak vyloučení.
~ Neighbor Unreachability Detection (NUD) – zjištění dostupnosti uzlu, dynamická síť, proto musí zjišťovat změny. Zjištění probíhá díky protokolu TCP, dostupnost souseda, pokud nepracuje musí uzel zjistit dostupnost sám. Několik způsobů, zašlu adresu a dostanu linkovou, je v síti. Pokud nedostanu, pak nemusí být dostupný tento uzel (je tu nějaký timeout). Máme tabulku sousedů, než odpoví (incomplete), pokud dostaneme linkovou adresu (reachable), dále (stale) musíme čekat až informaci dostaneme, (delay) timeout vypršel ale není to dlouho a (probe) zkouška, znovu.
~ Autokonfigurace – manuální konfigurace z IPv4 nechceme, chceme dynamiku. Dva typy: stavová DHCPv6 (odpovídá DHCP), používá se k obdržení dalších parametrů – informace o DNS serverech. Nová je bezstavová, generuje IPv6 adresy, předpokládá, že v síti jsou zařízení, které mají informace aby mohli být generovány adresy IPv6. V nějakých

časových intervalech v síti probíhá oznámení, kdy směrovač posílá parametry sítě. Když přistupuje nový uzel, který potřebuje adresu. Tak buď počká na směrovače, nebo si vyžádá výzvu směrovači a žádá informaci všechny. Adresu vytváří z informacích, které získal a ze své linkové adresy. Až ji vytvoří, testuje zda je jedinečná. Postup: nejprve generování linkové lokální adresy – potom test DAD – přiřazení této adresy – kontaktuje se směrovač – nastaví se adresa na směrovači aby o ní věděl. Příznak M = stavovou metodu, O = stavová konfigurace kromě adresy. Pokud má směrovač nastaveno lifetime na 0, nemůže být použit. Reachable time – jak dlouho lze považovat souseda za dostupného. ICMP options – source L2 adress, MTU, prefix information.




IPv6 – Mobility Support – myšlenka, že vždyky je někde doma/home. Použití: Home Address (unicast trvalá adresa) a Care-of address (globální mobilní adresa – když je jinde v síti). Correspondent Node (CN) – peer uzel se který komunikuje mobilní uzel a Home Agent (HA) – domáci routr, který je vždy dostupný, obdrží datagram a předá ho. Optimalizace není důležitá, komunikace může probíhat přes HA vždy. Ha může být statický i dynamický. Komunikace může probíhat i přímo. Exsitují dvě možnosti komunikace – bidirectional tunneling (obousměrné tunelování) nebo optimilizace trasy. Postup: 1. Home Test init (určí dom. Adresu) 2. Care-of Test init (určí péči) 3. reakce Home Test, lze generovat Home Keygen Token 4. reakce Care-of Test, lze gen. Care-of Keygen Token 5. Vygenerován 20bajtový klíč.
General Security Practices – důvěryhodnost, integrita, dostupnost, autentizace, autorizace, vedení (účetnictví), nezavrhnutí. Musí být zajišťovány dvě základní věci: šifrování, kontrolní součet, nejlépe obojí. Existují dvě formy šifrování – Secret Key Cryptography (symetrické šifrování) – odesílatele a příjemce se musí dohodnout na společném tajemství a Public Key Cryptography (asymetrické kryptografie, šifrování) – algoritmus používá pár klíčů, skládá se z veřejného a soukromého klíče. Message digest (hash) – funkce jedinečný výstup.

IPv6 – Security Support – IPsec je součástí protokolu viz RFC. Prvky IPsec: protokol pro autentizaci – AH (Authentication Header), protokol pro šifrování – ESP (Encapsulating Security Payload), definice pro použití kryptovacích algoritmů, definice pro bezpečností politiku a správu klíčů.
~ Security Associations (SA) – sada bezpečnostních informací, tři prvky – klíč, šifrování a mechanismus autentizace a další parametry, jednosměrné dohody → poskytovat šifrované a ověřené duplex komunikace, 4 SAs jsou nezbytné. SA je definován souborem tří parametrů: ~ Security Parameter Index (SPI) – 32-bitové číslo jednoznačná identifikace konkrétní SA, ~ IP Destination Address – adresa zařízení, pro něž je SA založena, ~ Security Protocol Identifier – určuje, zda je toto spojení pro AH nebo ESP.
Máme proto protokly pro automatické vyjednávání – Starý přístup: ~ Internet Security Association a ~ Key Management Protokol a ~ Internet Key Exchange verze 1 (IKEv1) – Současný přístup: ~ Internet Key Exchange verze 2 (IKEv2) zjednodušuje IKEv1, opravuje chyby, ale snaží se být stejný. IKEv2 – automaticky naváže SAs vytvoří/smaže materiál, ověřuje komunikaci. Dvě fáze: 1. bezpečný kanál pro vyjednávání a 2. bezpeční kanál pro přenos.




IPsec má dva druhy dopravy: (viz obr)
1. Transportní režim
2. Režim tunelu (navíc zapouzdření celého datagramu)

~ AH (Authentication Header) – protokol, který umožňuje autentizaci buď celého nebo části obsahu datagramu, provádí přidáním hlavičky vypočítané na základě hodnot. Kroky protokolu: 1.SA musí být nastaven mezi dvěma zařízení (nikdo další to neví) 2. Na zdroji AH provádí Integrity Check Value (ICV) a dává výsledek do spec. Hlavičky. 3. Cílové dělá kontrolní součet, zda se něco nezměnilo. Přítomnost AH ověřuje intergitu. AH je pouze autentizace.
~ ESP (Encapsulating Security Payload) – chrání data před neautorizovanou stranou, šifrování. Má podobné autentizační schéma jako AH. Nemáme jen hlavičku ESP header, ale i ESP trailer a ESP authentication data. Někdy je lepší AH někdy je lepší složitější ESP, proto je dobré mět i AH.

Kvalita služby QoS – IPv4 funguje tak, že pokud je možné jde co nejlépe – Best effort. Postupně vznikly dvě metody:
(1) Integrované služby – jsme schopni zajistit cestu a domluvit se na ní, po ustavení spojení sme schopni přeposlat data. Postupně vznikly některé protokly na dohody zpracování – např. Resource reSerVation Protocol (RSVP) nebo

YESSIR , v bezstavové síti zavádíme tímto stavy, což je špatné a komplikované – neujalo se.
(2) Diferencované služby – trošku plýtváme, máme def. třídy. Není tu stav, nikdo si nic nepamatuje, pouze směrovač co zvládá diferencované služby a když dojdou data určité skupiny, musí je přeposlat do třídy shodné či vyšší. Díky tomu se používají, jednoduché a bezstavové. V hlavičce se nachází:
~ Traffic Class – 1bitové pole, definuje 64 různých kódu, pole specifikuje jak bude paket zpracováván. 0=běžný paket. Poslední dva bity – speciální – jsou používány ECN – používány pro Congestion Notification, když je směrovač zatížen.
~ Flow Label – 20bitové pole, aby byl tok odbavován stále shodně, třeba u videa.

IPv6 Transition – Porting Applications – Jak konvertovat a navrhovat kód pro IPv4 tak i současně pro IPv6. Převádění mezi IPv4 a IPv6. URL reprezentace, problém může být i více adres na interface v IPv6. Navrženy tři možnosti pro koexistenci obou IP:
~ Dual stack – efektivita je nižší, musíme provozovat dva separátní protokoly, dva stacky. Problém s DNS, směrování... ale není to optimální řešení
~ Tunneling – např. bezpečností tunel viz předchozí text, někdo musí vytvářet hlavičky, držet tato data. Problém pokud vypadne ten kdo to má dělat. Výhoda – přímá komunikace – nevýhoda – ne všechno je stejné.
~ Translators – překládání adres – není vhodný

**[3] Směrovací mechanismy** – směrování je v L3, nespojené sítě s datagramy, nemáme informace o topologii sítě, potřebujeme pomocí lokálního směrování vyřešit směrování globální. Základním cíle je najít optimální (metrika) cestu. Dvě fáze: 1. Budování mapy 2. Posílání. Směrovač řeší jen na kterou připojenou linku paket přepošle → přiblíží paket k cíli a postupně se dostává do cíle, až se do cíle dostane (hop-by-hop). Reprezentace – teorie grafů (Bellman-Ford algorithm – jeden cíl and Dijkstra's algorithm – všechny cíle). Oba v centralizované i distribuované variantě. Internet je – distribuovaný – hop-by-hop – deterministický – single-path – dynamic path selection.
Směrovací protokoly:
~ Distance Vector (DV) – Bellman-Ford algorithm – Hlavní vlastnost je, že sousední směrovači si buď periodicky nebo při změně, vyměňují kompletní kopie směrovacích tabulek a provádějí změny cest. Veškerá informace o síti je posílána sousedům. Nevýhody – i špatné info se posílá všem. Použití je u malých sítí.
~ Link State (LS) – Dijkstra's algorithm – Směrovači si periodicky vyměňují informace, ne kompletní kopie, ale pouze o přilehlých linkách. Informace o mých sousedech všem. Větší množství kratších zpráv, více počítání pro směrovač.
~ Path Vector (PV) – modifikace DV – není jen cíl, ale i celá cesta v tabulkách, jednodušší detekce cyklů,

Autonomní systémy – internet se už nebral jako jedna síť, ale síť sítí. Odpovídají tzn. doménám 16bitová čísla, např. CESNET, PASNET atd., 3 druhy Stub AS, Multihomed AS a Transit AS. Potřebujeme aby byla zpráva směrována mezi dvuma AS, na hranici AS a pak druhým AS k cíly, směrování tedy interior (IGP) a exterior (EGP).

| Protocol | RIPv1 | RIPv2 | IGRP | EIGRP | RIPng |
|---|---|---|---|---|---|
| Address Family | IPv4 | IPv4 | IPv4 | IPv4 | IPv6 |
| Metric | Hop | Hop | Composite | Composite | Hop |
| Information Communication | Unreliable, broadcast | unreliable, multicast | Unreliable, multicast | Reliable, multicast | Unreliable, multicast |
| Routing Computation | Bellman–Ford | Bellman–Ford | Bellman–Ford | Diffusing computation | Bellman–Ford |
| VLSM/CIDR | No | Yes | No | Yes | v6-based |
| Remark | Slow convergence; split horizon | Slow convergence; split horizon | Slow convergence; split horizon | Fast, loop-free convergence; chatty protocol | Slow convergence; split horizon |

DV protokolová rodina:
~ RIP protokol – neustálá výměrna zpráv (30s-UDP), malé linky, není vhodný pro redundantní sítě, metrika 1-16. Nevýhody verze 1 – nekonečno = 16, není pole pro masku, není jak specifikovat adresu, problém s cykly. Proto verze 2, kde je navíc Route tag, subnet mask, next hop, lepší pro CIDR, navíc nepoužívá broadcast (v.1) ale muticast.
~ Interior Gateway Routing Protocol (IGRP) – vychází z DV přístupu, ale změna metriky, 5 druhů metrik, pro každou cestu. Pracuje nad IP protokolem, povoluje vícenásobnou cestu, dovoluje vyrovnávání sítě. Problém nepodporuje různou délku subnet masek – nepodporuje CIDR. Novinka MTU – nejmenší MTU sítě a metrika – komplikavanější, 4parametry, 5 vah. Problém různě nastavených směrovačů, když to má každý povinně nastaven.
~ Enhanced Interior Gateway Routing Protocol (EIGRP) – obnobný protokol jako IGRP, vylepšení s CIDR notací, ale dělán pro IPv4.

LS protokolová rodina:
~ Open Shortest Path First (OSPF) – nejpoužívanější LS protokol, metrika = cena = číslo, které přiřadí směrovač. Ke každému rozhraní máme danou cenu, žádné nekonečno. Dříve bez autentizace, v.2 s autentizací, v.3 IPv6 adresy, využití IPsec. Zavádí směrovací oblasti, jako další hierarchii = subdomény. Problém vyvažování sítě, load balancing. OSPF zprávy vnořeny přímo do IP adres. 5Druhů zpráv: Hello Packet, Database Description Packet, Link State Request Packet, Link State Update Packet, Link State Acknowledgement Packet.
~ Intermediate System To Intermediate System (IS-IS) – také pracuje se síťovou hierarchií, dvě úrovně. Oba používají:

hello pakety, podporují různou délku CIDR, LS – Dijkstra apod. Liší se: nad úrovní L2, využívá linkové rámce ne datagramy → snadnější adaptace v IPv6, je robustní. Metrika – také číslo, ale kratší. Méně rozšíření oproti OSPF, ale zas je více komunikativní, lépe škáluje, lepší pro větší sítě.

PV protokolová rodina:
~ Border Gateway Protocol (BGP) – způsob směrování mezi AS, současnost v.4, verze ukazují změnu internetu, přizpůsobování změnám z života, pracujeme přímo. Protokol si pamatuje přímo cestu do cílové sítě, usnadňuje odhalit možné smyčky. Umožnuje definici směrovacích pravidel, používá počet hopů – metriku a CIDR anotaci. Směrovači posílají zprávy přes TCP (port 179), oznámení se skládá z cílové sítě a atributů, které do cíle vedou. Záleží také na domluvě mezi operátory – cena za výměnu. Pravidla mají přednost před metrikou, až pak se metrikou používá nejbližší cesta. Zprávy: OPEN, UPDATE, KEEPALIVE, NOTIFICATION a ROUTE-REFRESH.
Základní problém, kam mají být všechna data zasílány – dvě část: interní a externí, musí o sobě vědět – IBGP protokol.

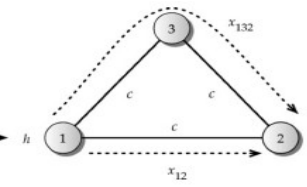
Architektury směrovačů – dedikované výpočetní stroje, dříve běžná pc s více kartami a sw. Protože jsme potřebovali větší výkon, tvorba velkých směrovacích strojů, přímo pro směrování. Směrovač musí: 1. směrovat a ohledávat okolí (výměna informací se svými sousedy, počítá nejlepší cesty – tabulky) 2. posílání paketů, rozhoduje na který interface paket pošle. Směrovač pracuje s IP hlavičkou, TTL Lifetime Control a kontrolní součet. Směrovač nejprve zkontroluje IP hlavičku, délku hlavičky a kontrolní součet, potom kontrola TTL, pokud změníme TTL, je třeba upravit kontrolní součet. Potom hledáme cestu – podle cílové destinace, která se srovnává s tabulkou na výstupu. Až se vybere víme kam jde a známe MTU, potom proběhne fragmentace na velikost odpovídající MTU. Další funkce je: klasifikace paketů, překlad paketů, priorita paketů, směrování protokolů (OSPF, BGP, RIP) , systémová konfigurace, management routrů.

Prvky routeru: Network Interface, Forwarding Engines (tabulky), Queue Manager (zajišťuje činost bufferu), Trafic Manager (odpovídá za prioritu, regulace rychlosti), Backplane (propojení jednotlivých interfaces, Route Control Processor (směrování tabulek a zanášení změn).

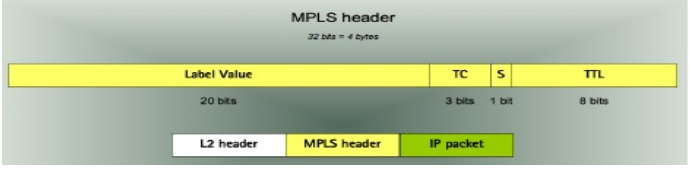
Vyhledávání cesty – problém u CIDR notací, proto vyhledáváme adresu s největším prefixem. ~ Lookup Speed – 50% paketů krátké TCP, 40bytes = 1 Gbps link ⇒ prefix lookup should not exceed 320 nanosec, 10 Gbps link ⇒ 32 nanosec, 40 Gbps link ⇒ 8 nanosec. ~ Memory Usage – větší přesnost = více paměti ~ Scalability – škálovatelnost – čím delší tabulku, tím delší čas zpracování, kapacita pro dodržování ~ Updatability – udržování tabulek.
~ Naive Algorithms – procházíme lineárně seznam O(n)
~ Trie-based Algorithms – stromový, více typů: Binary Tries, Multibit Tries, Compressed Multibit Tries.
~ Jiné způsoby – hledání podle délky, hodnoty, či novinka začínající se používat jsou hardwarové algoritmy.

Filtrování a klasifikace paketů – kvalita paketů, počítání prošlých dat na směrovači není snadné (pro operátory), standardizace tu zatím moc nefunguje. Další je prevence útoku směrovačů. Pravidla klasifikace paketů může být rozmanitá, nemusí žádné pravidlo nebo i více, dokonce i v rozporu s pravidly. Obvyklý způsob je ukládání v databází a každé pravidlo specifikuje, kdy se aplikuje a co se má dělat. Pravidla jsou pak seřazená a my hledáme nejrychleji to co použijeme. Pravidla, které se použijí dříve jsou dominantní nad ostatníma. Algoritmy: naivní (nepoužívá se), dvoudimenzionální, d-dimenzionální, rozděl a panuj, atd.

Dopravní inženýrství v IP sítích – problém směrování, pokud chcu posílat data, používám Shortest Path First (SPF), problém vytížení, všechno půjde nejlepší cestou – zatížení této části sítě. Potřebujeme tedy znát topologii, zátěž (pokaždé jinak, záleží na době, dni, lokalitě), provoz. dopravní kritéria. Jak často opitimalizovat – udává se asi den, je to náročné výpočetně. Ze sítě získávám data, topologie, měření provozu, vypočítáme váhy a pak určíme kama budeme posílat data. Řešení je ohodnocení hran tak, aby nedocházelo k zátěži na některých uzlech – jak to vyřešit, jak správně nastavit, některé cesty mohou být dlouhé. *Když se přetíží směrovač, tak háže pakety do bufferu.* Simple Network Management Protocol (SNMP) – nejpoužívanější, také sondy NetFlow atd. U velké sítě musíme počítat váhy dynamicky – používáme OSPF a IS-IS. Jak se počítají váhy? (h celkem toku, x12 + x132 = h, nelze -) Potřebujeme vypočítat optimální cenu c. Total cost = ξ12 x12 + ξ132 x132. Nerovnice x12 <= c, x123 <= c a x12 >= 0, x123 >= 0. Naším úkolem je minimalizovat total cost. Navíc ještě potřebujeme load balancing (minimalizace zátěže) a avarage delay (průměrně zpoždění). Ve skutečném internetu je více přenášených dat z více uzlů > složitější nerovnice.




Multiprotocol Label Switching (MPLS) – patří do L2 i L3, směrování podle labelů – tabulek, zrychlení provozu, zjednodušení směrování. Máme hraniční (směrování a cesta) a vnitřní (pouze rozeznávají značky paketů a posílají pakety podle cesty – značek) směrovače. Zavádíme defakto spojení v MPLS cloudu, řada

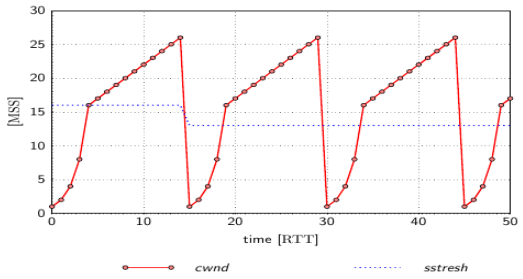


paketů se stejnou cestou zpracováváme stejně i tak je efektivnější. Obousměrná = dvě jednosměrné cesty. MPLS se opravdu používá i na páteřních sítích. Paket vstupuje do MPLS cloudu, je udělána jeho klasifikace, podle cílové destinace se zařadí pakety do tříd – Forward Equivalence Class příp. jinak. Vytvoření značky, vytvoření cesty, distribuce značky, na směrovačích tabulky, značky se mohou měnit, přeposílání paketů. Paket zapouzdří MPLS před něj dá značku. Není unikátní label ani stálý, může se měnit ze směrovače na směrovač, proto tu máme tabulky abychom to rozeznali. Navíc si pamatuje, které použil. Edge Label-Switched Routers (Edge-LSRs) – hraniční – vstupní a výstupní (TTL zmenšení) a Core Label-Switched Routers (Core-LSRs) – vnitřní (fordwardovat paket, přeposílat paket, modifikace labelů). Hlavička TC = Traffic class a S = Stack. Jeden paket může mít více labelů (značek), přidávají se hlavičky, třeba MPLS s VPN, nebo s Traffic Engineering se dvěma značkami apod. Oproti směrování, musíme rozšířit značky do celé sítě, protoly můžeme rozšířit, třeba BGP, RSVP, LDP, TDP atd. Posílání značek = dat – TCP i UDP. MPLS nám dává možnost pracovat s toky,
~ Generalized MPLS – princip MPLS rozšířen do Time-Division Multiplexing Capable, Packet-Switch Capable, Fiber-Switch Capable nebo třeba Grid-enabled GMPLS (nepoužívá se).

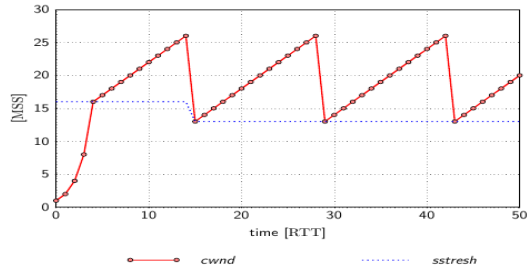
QoS-Based Routing – jak by mělo vypadat směrování podle QoS: *vědět informaci o požadavcích uzlů *optimalizaci využití síťových zdrojů *degradovat výkon. V běžné síti je velmi složité získat informace a udržet QoS. Jak často získávat dodatečnou informaci pro udržení QoS. Škálovatelnost – vytváříme hierarchie sítě pro zjednodušení. Administrace – Diffserve – dělení do tříd. Integrace – QoS-based routing and Best-effort routing – jejich koexistence.
~ Routing Algorithms – *Source-based routing algorithms – musíme dobře znát lokální rozdělení, nejjednodušší možnost směrování, musí být ale malá. *Hop-by-hop routing algorithms – postupné přibližování, nejlepší pro sítě internet, musíme doplnit informaci o parametrech sítě, neznáme cestu – minus zpoždění. *Hierarchical routing algorithms – směrovače rozdělíme, agregujeme, vhodné pro velké sítě, výhodné pro QoS, něco mezi hop a source.
~ Private Network-Network Interface (PNNI) – zástupce hierarchického směrování pro ATM sítě. *ATM – pokus o hibrit v 90letech o spojovanou a nespojovanou síť. Tenkrát výhodné, tou dobou lepší jak ostatní. IP běželo v ČR 4-5l na ATM.*
~ QoS routing extensions to OSPF (QOSPF) – rozšíření.

**[4] Traditional TCP** – tvoří drtivou část dnešního provozu, poskytuje zabezpečení provozu – aplikace se může spoléhat na doručení a na pořadí doručení. V TCP nedochází k zahlcení, kontrola. Agresivita – schopnost využít všechny možnosti, responsivnost – rozšíření/snížení okna, férovost. Problém – použití TCP pro přenos HD videa. Problém – flow control vs. congestion control. Kontrola zahlcení – klasické okno AIMD, u *Tahoe (zvětší o 1 a za každou strátu se nastaví další hodnoty) a *Reno (retransmise pokud dojde ke ztrátě a narazí se na další 3 pakety, dojde k rychlému přeposlání. Lepší využití pásma je u Rena. TCP *Vegas – se snaží předcházet ztrátě paketům, > začne zmenšovat okno. Pokud dojde ke ztrátě paketů: Tahoe (nutno přeposlat aktuální množství dat), Reno (stačí přeposlat jeden segment), NewReno (nutno poslat více segmentů, vhoda méně ACK) nebo Selective Acknowledgement (SACK) (stačí 1 paket). *Response Function *Férovost – u TCP zajištěna dobře. ~ Multi-stream TCP – zajistíme si více streamů, lineární nárůst podle počtu streamů, nevýhoda – zahlcení sítě se nedotkne jen jednoho streamu, ale všech. ~ TCP implementation tuning – další možností je vylepšení implementace, *přesun také na HW, *zero copy přístup (snaha oprostit se od kopírování – implementován na linuxu), *Web100 (monitoring, ladění a podporu)

Traditional TCP – Tahoe II.




Traditional TCP – Reno II.



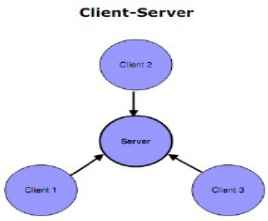

~ Conservative Extensions to TCP – *GridDT – ad-hoc modifikace, *ScalableTCP – AIMD → Multiplicative Increase Multiplicative Decrease (MIMD) – pokud je ale okno menčí než nějaký parametr pak zase AIMD, *High-Speed TCP (HSTCP) – vychází opět z přepínání AIMD/MIMD (na základě pozorování ScalableTCP), využívá speciální parametrizaci congestion window, *H-TCP – agresivní TCP, rychlý růst okna, snížena férovost, kvadratická složitost, *BIC-TCP – tento algoritmus je používán v linuxovém jádře, dojde k rychlému nárůstu, pak konstantní a pokud vše v pořádku tak pokus o rychlejší,, 4 fáze – (1) výpadek (redukce okna), (2) aditivní nárůst (polovina min max je velká, snižuje se, používá se spíše nárůst o konstantu, která roste), (3) binary search (dosáhneme poloviny min max), (4) hledáme maximum (2-3), přenos dat bude relativně plynulý.

~ Quickstart (QS)/Limited Slowstart – proč zasahovat do L3? Navržená modifikace – rychlejší informace od směrovačů, co se děje na síti, jak vypadá stav sítě. *E-TCP – Early Congestion Notification (ECN) – nastavuje bit, jak vypadá stav bufferu (mohou ho měnit i směrovače na síti) – nevýhodou je, že směrovači musí umět pracovat s tímto E-TCP. Příliš se nepoužívá. *FAST – pomocí ECN bitu, rychlejší reakce.

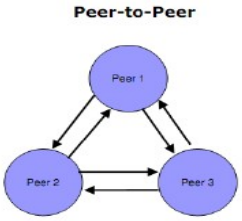
~ Přenosové protokoly – *Tsunami – pracuje s TCP a posílá negativní ACK, vlastní data přes UDP. Zajištění přenos a lépe využíváme linky. *Reliable Blast UDP – RBUDP – posíláme kopii dat a jsme schopni udržet tuto kopii, UDP data, TCP spojení co nedorazilo. *eXplicit Control Protocol – XPC. *STCP (orientace na zprávy UDP, zajišťuje doručení), *DCCP, *STP (jednoduchost, ale horší vlastnosti), *Reliable UDP (přidáno zajištění spojení, pro IP telefonie), *XTP.

Současný stav – multistream TCP, post-TCP a pomalu si rozšiřují rozšiřující, pomalu se prosazuje agresivní metody – ale ne pro normální sítě. Nezpracovávají se pakety ale toky!

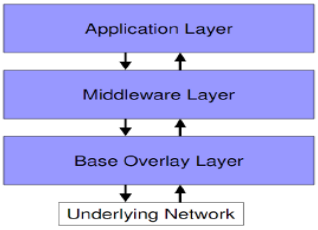
**[5] <u>Peer-to-Peer (P2P) sítě</u>** – aplikační úroveň! Distribuované aplikace – skládá se z více SW modulů, které jsou umístěny na různých výpočetních jednotkách a jsou propojeny komunikační sítí. Jednotky navzájem komunikují a předávají si informace a synchronizují se. Pro komunikaci máme dvě možnosti: Client-server a Peer-to-Peer (+ hybrid).




~ Client-server – centrální, sekvenční či paralelní, klient čeká a poslouchá, pokud je zaměstnán pak řazení do fronty, připojení serveru nesmí být úzkým místem. Vždy klient-server nikdy klient-klient. Selhání klienta není selhání celého systému. Př. SSH, web server (prohlížeč, aplikační klient), FTP klient.




~ Peer-to-Peer – každý může být klient a zároveň server, mohou být nezabezpečené → selhání jednoho neznamená selhání všech, má jednu důležitou vlastnost – přirozená škálovatelnost (přidáváním zdrojů, roste množství zdrojů), samoorganizace, neexistuje globální architektura. Hlavní vlastnosti: symetrické role (server i klient), velmi dobrá škálovatelnost, heterogenita (paměť, výpočetní výkon – každý může přispívat různě), decentralizace, dynamika (připojování – odpojování), zdroje sdílení, samoorganizace.
Srovnání: vývoj, tvorba C-S, řízení C-S, škálovatelnost P2P, bezpečnost C-S, spolehlivost C-S.
P2P má tři vrstvy (nemusí jít vždy dobře rozlišit):



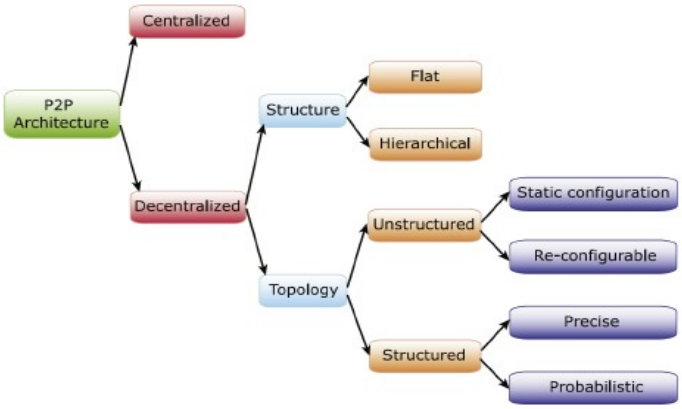

~ Base Overlay Layer – (základní překryvová úroveň) – nacházení nových uzlů, podpora a posílání zpráv mezi P2P. Pracuje jako virtuální stack, který nakopíruje info o síti do L5 a L6, z vlastní fyzické sítě. Pracuje pak tak, že se mapuje na tuto síť. Překryvová síť neřeší jak se pojí dva body, ale bere je jako spojené na přímo. Problém, že máme vlastně dvě sítě a dva různé počty hopů (mohou se lišit, což není tak dobré).
~ Middleware Layer – bezpečnost, zacházení se zdroji, shlukování uzlů do skupin, obdobně jako middleware. Existují frameworky.
~ Application Layer – dodává přímo aplikaci a propojuje ji. (př. Skype. SETI@home atd.)

<u>Překrývání a Peer Discovery –</u> virtuální síť nám usnadňuje práci, získání informací. Není nejjednodušší vytvořit síť, někdy lepší *statická konfigurace (ta však snižuje přirozenou škálovatelnost), řešení – je navržena podmnožina statická, ostatní autokonfigurace. *Centralizace – každý Peer získá info, není nejvhodnější. *Propagační techniky – není nutné znát všechny členy sítě, stačí se přihlásit k někomu, získá repliky tabulek a má startovací bod a potom už automaticky upravuje sám tabulky. Tři základní topologie:
~ Náhodný Mash – Dobře funguje pokud stupeň uzlů je +/- shodný, je lepší propojovat navzájem blízké uzly.
~ Tiered Structure – sdružení do určitých oblastí navzájem propojených, komunikace se vede na několika málo uzlech, úrovní může být několik (většinou jen pár), př. Kazaa, lepší vyhledávání, připojení je složitější, *Vždy když máme hierarchii, je snazší vyhledávání, ale platíme za přidávání do hierarchie. Vyhledáváme třeba podle indexu.*
~ Ordered Lattice – používá se, využívá se informace lokality, využívají CAN a DHT. Mřížka může být 2 ale i více dimenzionální, horší nalezení místa pro vložení nového uzlu – přeorganizování.
Vlastní vyhledávání – u centrolizoaných získá lokalizaci uzlu nebo přímo data, P2P – dotaz můžeme poslat záplavou, pomocí hierarchie vyhledáme apod. Nezískáme jestli tam data jsou nebo ne, ale jen lokalizaci.

<u>Taxonomie P2P</u> – rozdělení
~ Centralizované – kombinace C-S a P2P, mají 1+ centrálních prvků, dotaz uzlu na centrální uzel a ten se spojuje pak s uzlem, pouze usnadňuje lokalizaci. Nevýhoda – úzké místo, centralizace – výpadek, přetížení. Přesto využívaný – SETI@home, jabber atd.
~ Decentralizované – „čisté P2P“, nemáme žádnou centrální autoritu → nutný vyhledávací algoritmus (časově

náročnější), dobrá škálovatelnost, robustnost, klasické vlastnosti P2P. Př. Gnutella, Crescendo, PAST, FreeNet, Canon atd.

Dále dělíme podle struktury: *ploché a *hierarchické a podle topologie na *strukturované a *nestrukturované. U *nestrukturovaných je každý uzel zodpovědný za své data, pomůžou sousedé, špatná lokalizace, nemáme jistotu získání kompletní odpovědi, náhodná procházka. Špatné objevení položky, ale snadné budování. *Strukturované – řízeno distribuovaně, máme zajištěnu strukturu, musíme věnovat značné prostředky na budování sítě. Máme také *hybridní sítě.

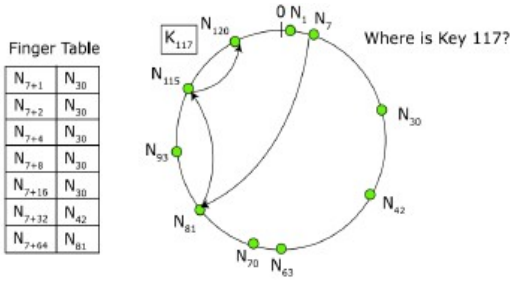



Směrování v P2P – neplést si se směrováním na úrovni L3, šíření dotazů (neznáme cílovou adresu). Hlavní problém: máme klienta který chce data z nějaké sítě, ptá se sítě, ta si předává data, až dotaz dojde k někomu, kdo data vlastní a ten posílá odpověď (nestrukturovaná). Pokud máme centralizovanou – strukturovanou, pak se ptá centrálně jednoho uzlu (DB), tato entita může být ale úzkým místem systému. Srovnání mechanismů – paměť (rozsah, čas vyhledávání, záplava), efektivnost, použitelnost (typy dotazů), pokrytí, škálovatelnost (nejzásadnější metrika).

~ Nestrukturované – každý uzel obsahuje data, musí mět linky k několika sousedům, když se chce spojit okopíruje si sousední uzel, používají se techniky na základě záplavy, mají TTL aby se nešířily nekonečně dlouho. př. Breadth-First Search (BFS) – Gnutella, Depth-First Search (DFS) – FreeNet, Heuristic-Based Routing Strategies.

Heuristické směrovací strategie – každý z uzlů si drží historii aby věděl od koho došly dotazy, tyto statistiky slouží pro pozdější vyhledávání a dotazování. Nevýhodou statistiky můžou být nepřesné, nevíme moc informací – nezahrnují obsah dotazu. Pokrokem je *inteligentní vyhledávání, pamatují si i obsahy dotazů. Dalším vylepšení jsou *lokální přímé indexy. Pokud je k=0 BSF, více úrovní, uzel reprezentuje nejen sebe ale rozsah uzlu o k. Výhodou jsou redukujeme výpočetní náročnost, díky zpracování na menší počet uzlů, nevýhodou – režie na uzlech, nekonzistence, náročnost. *Náhodné procházky – uzel který obdrží dotaz náhodně přepošle dál, dokud není nalezena odpověď. (hledání v kupce sena). Vylepšení je *k-náhodná procházka – k kopií, další pak už jen jednu kopii, zvětšíme počet dotazů k-krát, zvýšíme pravděpodobnost nalezení, lineární nárůst. Podobné k-náhodné procházce je také *Random Breadth First Search (RBFS) – navštívíme více uzlů, máme ale větší počet zpráv, exponenciální nárůst. *Adaptive Probabilistic Search (APS) – kombinace náhodného a pravděpodobnostního, každý uzel si udržuje tabulku, dva přístupy: optimistický x pesimistický (x možnost také přepínání mezi oběma). Je potřeba udržovat tyto pravděpodobnosti, více místa na uzlech. *Interest-Based Shortcuts – zkratky, snaží se vnést do vyhledávání informaci o obsahu, spojují uzly s podobným obsahem, když přijde dotaz →uzel může využít a přeposlat dotaz uzlům s podobným dotazem (s typem obsahu). Problém se zkratkama – budování zkratek – DB, rozsáhlé systémy problém rozsahu zkratek.

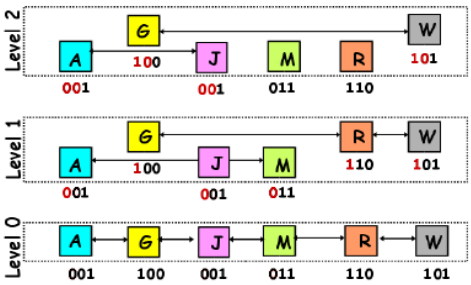
~ Strukturované – efektivnější, dělení: (1) systémy využívající hašovacích tabulek – Chord, CAN, Tapestry and Pastry, Viceroy and Crescendo, atd. (2) využívající skip listy – Skip Graph, SkipNet a (3) systémy využívající stromové struktury – P-Grid, P-Tree, BATON atd.



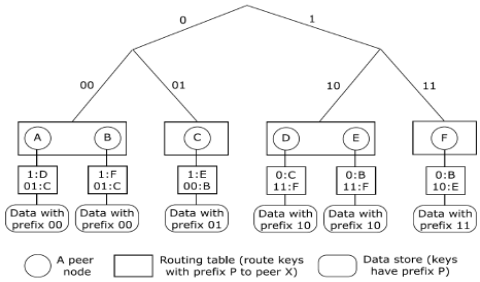

(1) Systémy hašovacích tabulek – každý uzle má svou část hašovací tabulky, vyhledáváme podle klíče. *Chord – nejjednodušší, identifikátor IP adresa, zahašuje se také metadata, uzly jsou umístěny na kružnici, datové položky musí být uloženy na uzlech (položka data s číslem menším než je haš umístíme na větší haš uzlu – data 12 budou na nejbližším hašu uzlu třeba 15 atd.). Vyhledávání pokud je větší pošle se dál, konec nalezení dat nebo haš je menší než haš uzlu kam byl přeposlán. Byly vymyšleny další mechanismy vyhledávání na Chordu je také škálovací vyhledávací algoritmus – ke každému uzlu není jen ukazatel následovníka, ale i další uzel, zkratky mocniny 2), tzv. přeskakujeme uzly. Problém budování! *Content Addressable Network (CAN) – 2+ dimenzionální plocha, uzel musí zpracovat všechna data, kam hašově patří a data spravují. Pokud pak hledá data, pomalu prochází přes uzly jednotlivých oblastní, rychlé vyhledávání. Budování je opět složité, je třeba id zahašovat, zjistit kam patří a musíme mu dát tuto oblast a přesunout data o které se má starat. Nesmí být oblasti které nikdo nespravuje, po odchodu zase začlenění oblastí. *Pastry (pracucí s prefixy) *Tapestry (sufixy) – nezávisle na sobě vznikly obdobné systémy – číslice se dají

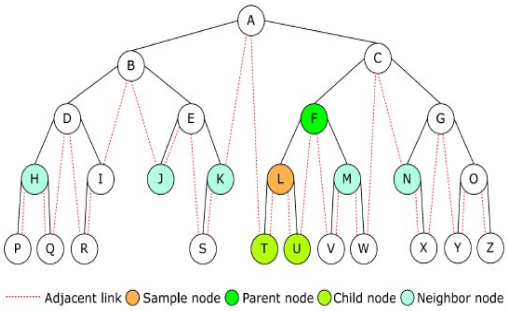
rozdělit na 2^b – konfigurovatelný parametr. Datová položka je zahašována, identifikátor nejbližší prefix/sufix co nejdelší stejný. Vybíráme takový uzel, který je datové položce nejblíže. Každý uzel má svou směrovací tabulku, řádky (délka prefixu), sloupce (další rozlišení). Další rozšíření leaf set. Jelikož je to dynamické, tak pokud nenalezeme data tak posíláme dalšímu uzlu podobnému. Problém budování systému.

(2) Systémy skip listy – *Skip list – pomocí setříděného seznamu položek, buduje se ve více úrovní, které se postupně zmenšují, až zbude jen jeden. Vyhledávání tedy lineární a pak horizontální. Cena je logaritmická, vyhledáváme shora dolů, nevýhodou je vyhledávání začíná na hoře, vždy na nejvyšším uzlu – na stejném uzlu. Snaha o řešení *Skip Graf – v každé vrstvě stejná počet uzlů. Na každé úrovni více seznamů, každý uzel začleněn ve více seznamech. Logaritmický počet úrovní, nejvyšší úroveň propojení uzlů se 2 spol. atributy, další 1 spol. a nakonec všechny propojeny. Uzel nalezen nejhůře na poslední úrovní. Přidávání odspodu, pomalu se přidává do seznamů. *SkipNet – zakruhované seznamy, kombinace kruhu a skip grafu.




(3) Systémy stromové – *P-Grid – nad uzlama postavíme virtuální strom. Pokud souhlasí vyhledá data, pokud ne podívá se do tabulky a najde podobný. Složitost – log2n, struktura nemusí být vyvážená, proto modifikace→ *P-Tree – b+stromy, *BATON – nebudujeme strukturu virtuální, ale strukturu stromovou, je doplněna dvěma dalšími linkami, které zabrání neustálému procházení přes kořen stromu. Viz obrázek, zná rodiče, děti a sousedy 2^i (ne všechny, protože by toho mohlo být mnoho a zbytečně). Při vyhledávání pak používáme známé uzly.




~ Hybridní – dva typy uzlů, obyčejné uzly a super uzly. Uzly jsou pospojované a nad další úrovní jsou pak spojeny super uzly. Př. KaZaA, BestPeer, Edutella (hyperkostka) atd.

| | structured P2P | unstructured P2P |
|---|---|---|
| routing | based on a routing table | flooding, random walk, ... |
| lookup possibilities | based on keys only | possibility to ask more compex queries |
| existing item is always found | yes | cannot be guaranteed |
| critical part | node join/disconnect | lookup/routing |

**[6]** **<u>Ad-hoc sítě a senzorové sítě – MANETs</u>** – (L2 a L3 pro uzly senzory a mobilní počítače)

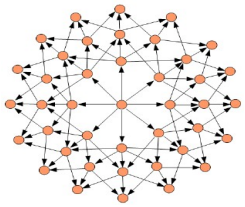
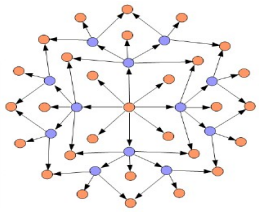
<u>Bezdrátové sítě</u> – pro pohyblivé uzly, infrastruktura – GSM, UMTS, WLAN atd. Co když tuto infrastrukturu nemáme, používáme ad-hoc sítí. Stále ještě ve vývoji. Ad-hoc sítě je skupina sítí, které komunikují mezi sebou a provozují komunikaci bez ústředního bodu, každý uzel je koncovým uzle i směrovačem. Topologie je dynamická, možnost změny polohy i přidávání/odcházení nových zařízení. Je potřeba podobného systému jako P2P. Výhoda nemusíte investovat do spojení, síť je odolná (vyřadíte jen část), lépe využíváme pásmo. Nevýhody – musíme využít principu autokonfigurace, omezenost dosahu, dynamika (směrování podle zastaralých dat). *Mobile Ad-hoc Networks (MANETs). Musíme upravit L2 a L3 na Ad-hoc sítě. Bezdrátové, spolehlivost nižší, odposlouchatelná, přesun častý, fyzická struktra odpovídá logické, *Vehicular Ad-hoc Networks (VANETs) – pohyb aut po silnicích. *Bezdrátové senzorové sítě (⇒ Wireless Sensor Networks (WSNs)) – nemusí iterovat s člověkem, ale s ostatními. Máme mnoho možností uplatnění, třeba i senzory v mostech, sopky, vlny, vlhkost na poli – zavlažování apod. Typické aplikace – proti lesním požárům, inteligentní budovy, zemědělství apod. Sledujeme energetickou náročnost, baterii, paměť, rozmístění, periodu vysílání, spolehlivost, vysílání, robustnost atd.

<u>Medium Access Control</u> (MAC) – v ad-hoc a senzorových sítích – koordinace uzlů, minimalizace kolizí pokud dva uzly vysílají současně. Nějaké kolize být můžou, ale nemělo by jich být hodně. Požadujeme vysokou propustnost, málo chyb, ale navíc i efektivitu využití energie! Kolize, přeslechy, vysílání, poslouchání – plýtvání energií. Navíc musíme počítat s mobilitou. Klasifikace:
*Soupeřící protokoly s rezervačním mechanismem – rezervace šířky pásma, synchronní x asynchronní, MACA/PR.
*Soupeřící protokoly s plánovacím mechanismem – měří jen někdy (šetření energie), LEACH, SMACS, TRAMA.
*Soupeřící protokoly – dva typy: sender-initiated protocols (ten kdo vysílá) a receiver-initiated protocols (ten kdo přijímá).
Problémy: *skryté terminály (nevidí na sebe, neví kdy vysílat), *blízkost terminálů – navzájem se blokují i když by

komunikace mohla probíhat – vyhranění si pásma pro sebe.
~ Busy tone protocol – ten kdo chce vysílat, vysílá a vyhrazuje si médium pro sebe, nelze signalizovat a vysílat v jednom signálu. Potřebuju dvoje vysílání na přenos pro jedny data. Není efektivní.
~ Busy tone multiple Access (BTMA) – velice špatné využití pásma, nepoužívá se.
~ MACA – Multiple Access Collision Avoidance – používá dvě zprávy, kdo chce vysílat tak Request to Send (RTS), potom nastaví Clear to Send (CTS) a začne vysílat, po konci potvrzuje doručení.
~ MACA – Energy Consumption Reduce – Power-Control MAC (PCM) – opět RTS/CTS dělají se na maximální úrovni, pak množství energie na přenos už jen DATA/ACK, kde šetří energií.
~ Sensor-MAC (S-MAC) – snaží se šetřit energií, jednotlivé uzly nevysílají a neposlouchají stále, ale jen občas s periodou, která se mění. Sleep/Listen. Listen = synchronizace/RTS/CTS.

Ad-hoc směrování – základní je šetření energií, čas od času spí/vysílají, nemáme infrastrukturu, musíme směrovat v souladu s MAC protokolem (přeslechy, kolize) a souladu s časovým plánem, většinou uzavřená síť, jak tedy směrovat? (1)Address-based routing – uzly mají speciální adresy – shodné v rámci ad-hoc sítě, cílené směrování. (2)Data-centric forwarding – pokud však žádné adresy nemáme (speciálně u WSNs), pak směrujeme podle obsahu dat a typu zprávy.
(1) Address-based Ad-hoc Routing – *proaktivní x *reaktivní (jen když je třeba dřív ne, zpoždění, ale úspora), table-driven x source-routing, ploché x hierarchické (speciální uzly s více pravomocemi, efektivnější) protokoly, lokalizované (GPS) x nelokalizované – nejsou disjunktní.

*Proactive Ad-hoc Routing – table-driven (mezi sousedy), link-state (mezi uzly, může být přeposlán kdekoliv).
~ Destination Sequence Distance Vector (DSDV) – inspirován RIPem, Belmanford, periodická výměna tabulek mezi uzly, je možné vyměňovat jen část tabulek, každý uzel ví, kde se nachází každý uzel sítě – velké tabulky! n*n
~ Optimized Link State Routing (OLSR) - uzly si samy počítají tabulky, záplavy – dvou typů uzlů, veškerá komunikace směrována na tento typ uzlů. Přístup efektivní, vhodné pro rozsáhlé sítě. Problém muticast, bezpečnost. (obr. záplavy a MPR záplavy).

*Reactive (On-Demand) Ad-hoc Routing – Negativum – hledání cest se používá záplav, i když až na vyžádání. Dynamika, než to zjistím, tak může být mimo síť nebo jinde.
~ Dynamic Source Routing (DSR) – dvě typy zpráv Route Request (RREQ) a Route Reply (RREP), ten kdo chce vysílat, zaplavý RREQ, jakmile dosáhne cílového uzlu ozve se RREP (a odešle uzlu posílajícímu) a pak se může přenášet. Vhodný pro menší bezdrátové sítě. Není to nejefektivnější.
~ Ad Hoc on Demand Distance Vector (AODV) – vylepšení DSR, používá stejné zprávy, ale zaznamenává celou cestu k sobě a posílá zpět, tím se zrychlý přenos.
~ Dynamic MANET On Demand (DYMO) – redukce režije ustavení cesty AODV
Geographic Routing – může se používat, GPSR, DREAM / Energy-Aware Routing Protocols – šetření a optimalizace. Mobilní mají čím dál větší význam a ještě se rozvýjejí. Vědět rozdíly mezi P2P a ad-hoc směrovnání.

**[7] Počítačová síť a multimédia** – informace složeny z několika typů dat a dány dohromady. Typy: text, obrázky, audio, video, interakce (vjem) nebo vůně. *text – FTP, HTTP, SMTP, protokol TCP bez chyb, někdy UDP ale s chybami. Většinou nemusíme řešit zpoždění. Kódování huffmanovo, LZW, Shannon-Fano... *audio – vlnění, digitalizace (vzorkování a kvantování), ztrátovost není hrozné, většinou nepostřehnutelné. 1-2% ztrát nemusíte postřehnout., větší citlivost na zpoždění, online přenosy, jitter – rozptyl. *grafika/animace – tok obrázků, různé koprese, podobnost s textem. *video – sekvence obrázků, 24-30obr/sec, obdobně jako audio, digitalizace, největší množství dat, HD 1,5Gbps, v lékařství pak ještě více, formáty (MPEG, H.261 atd.). Můžeme rozlišovat: media v reálném čase a nereálném čase, spojitá a nespojitá, tolerantní ke zpoždění a netolerantní.

Multimediální Požadavky na komunikační síť - multicast, mobilita, očekávají se nové protokoly pro přenos multimediálního obsahu, přenosy v reálném čase.
~ Packet Processing Delay - zpracování u odesílatele, přenesení, konverze dig/ana, zpoždění kódování,
~ Packet Transmission Delay - čas ztrávění na aktivní prvcích při vyhledávání, způsobené přenosem
~ Propagation Delay – propagační zpoždění – přenosové linky, vzdáleností více jak 1000km (200km = 1ms)
~ Routing and Queuing delay – směrovací, nejlépe tedy bez směrovače napřímo

*šířka pásma – TCP IP není schopno využít šířku pásma, *chybovost – vždy bude, minimalizace, oprava chyb na straně vysílajícího i přijímajícího – metody zálohy v horší kvalitě, metoda prokládání.
Real-Time Transport Protocol (RTP) – synchronizace hlasu a videa...
- Muticast – (konec poslední přednášky zbytek asi samostudium, na testu to nakonec asi bude tak pozor)