

## Úvod do počítačových sítí.

**Definice:** jako počítačovou síť označujeme skupinu počítačů, které jsou vybaveny technickými prostředky pro komunikaci, propojeny mezi sebou navzájem a s terminály uživatelů přenosovými kanály a vybaveny programově tak, aby zajistily komunikaci libovolného uživatele s programem na libovolném počítači, ale také dvou libovolných uživatelů mezi sebou, nebo dvou libovolných programů mezi sebou, při současném efektivním využití přenosových kanálů a vysoké spolehlivosti komunikace. Původně byly počítačové sítě určeny ke spojení vzdálených počítačů prostřednictvím datových spojů, které využívaly hovorové kanály telefonní sítě. Sítě tohoto typu se opírají o polygonální topologii, ve které jsou jednotlivé počítače spojené dvoubodovými spoji.

Spojení počítačů v těchto sítích lze zajistit přepojováním kanálů, zpráv nebo paketů.

1. **Přepojování kanálů:** Spočívá ve vytvoření fyzického datového spoje mezi koncovými účastníky. Nejvíce odpovídá telefonní síti.

- Výhody: malé, přesně definované zpoždění přenosu informace
- Nevýhody: malá přenosová rychlost

2. **Přepojování zpráv:** Přenos probíhá po zprávách. Zpráva je předána nejbližšímu sousednímu uzlu. Ten ji uloží do vyrovnávací paměti, analyzuje adresu adresáta a odešle nejvýhodnějším směrem dál.

- Výhoda: Maximální využití sdílených přenosových kanálů
- Nevýhoda: Přenos dat proměnné délky a z toho plynoucí problémy s přidělováním paměti v uzlech

3. **Přepojování paketů:** Modifikace metody „Přepojování zpráv“. **Paket** – fyzický blok dat, který má pevně stanovenou nebo shora omezenou délku a jehož prostřednictvím síť přenáší zprávy větších délek.

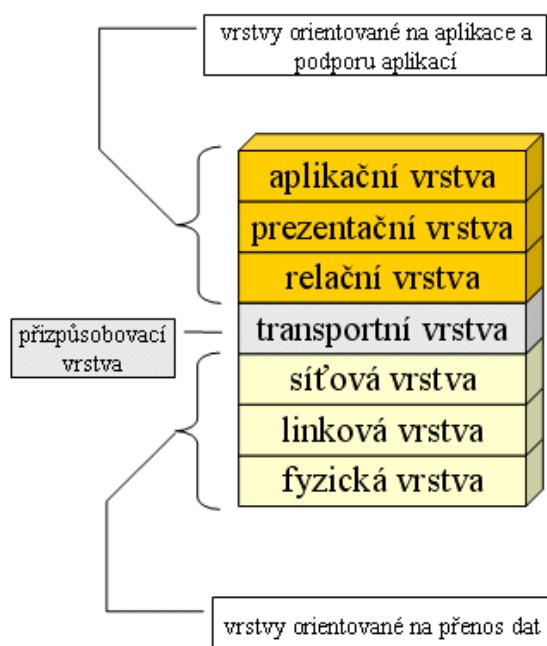
- Výhoda: Zjednodušení práce uzlů a zlepšení vlastností sítě.
- Nevýhoda: Nutnost rozkládat zprávy na do paketů před odesláním a po přijetí opět skládat.

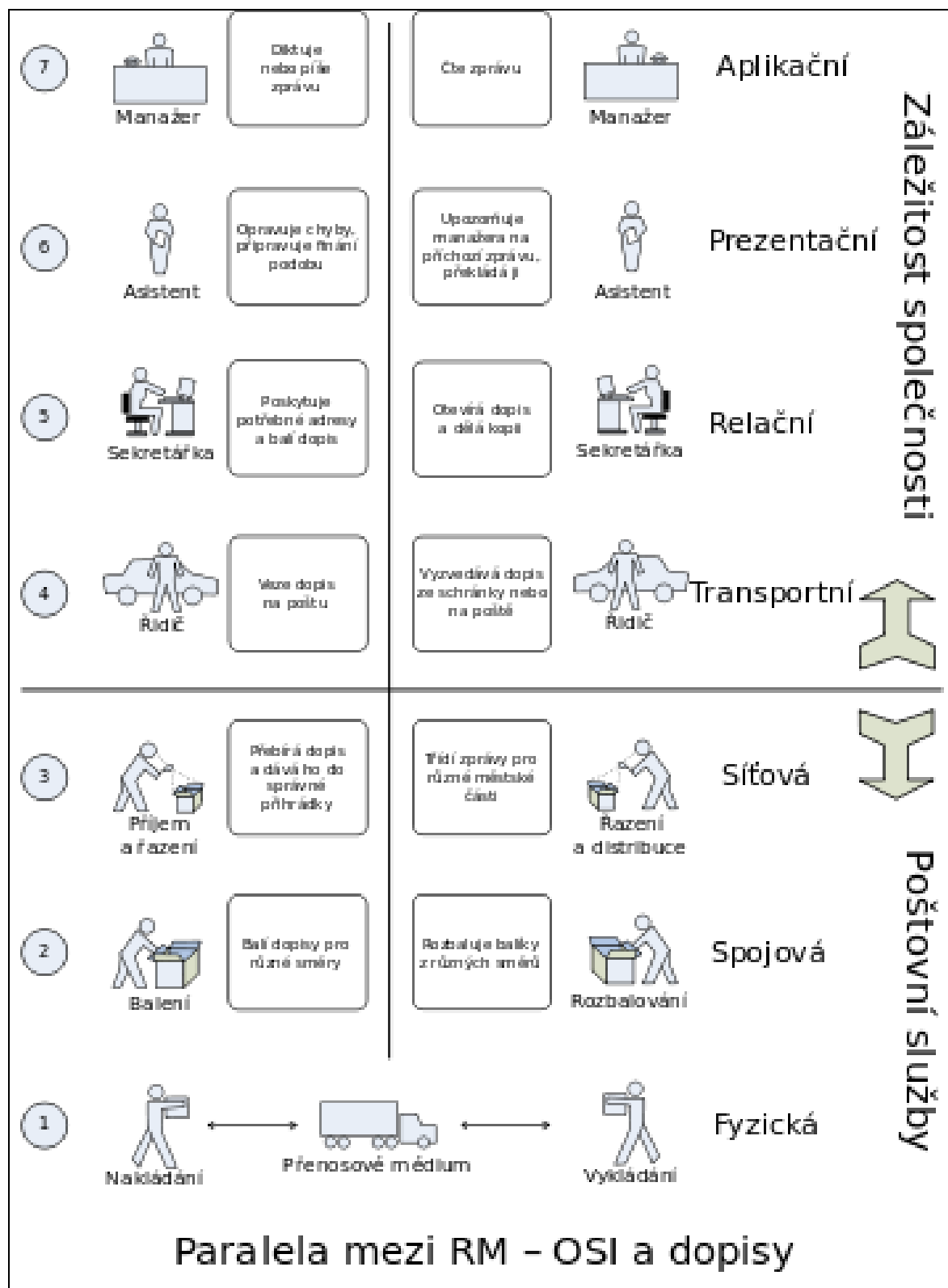
## Struktura programového vybavení počítačových sítí.

Programové vybavení sítí využívá fyzického média pro přenos dat. Toto je rozděleno podle doporučení **ISO/OSI modelu** na 7 vrstev. Úlohou referenčního modelu je poskytnout základnu pro vypracování norem pro účely propojování systémů.

Otevřený systém podle tohoto modelu je abstraktním modelem reálného otevřeného systému. Norma tedy nespecifikuje implementaci (realizaci) systémů, ale uvádí všeobecné principy sedmivrstvé síťové architektury. Popisuje vrstvy, jejich funkce a služby. Nejsou zde zařazeny žádné protokoly, které by vyžadovaly zbytečně mnoho detailů.

V praxi je model využit pro programování jednotlivých součástí síťového subsystému v modulech, které reprezentují jednotlivé vrstvy a komunikují mezi sebou pomocí rozhraní (API). Díky tomu je možné jednotlivé části snadněji naprogramovat a nezávisle nahrazovat (například vyměnit síťovou kartu, ovladač, aplikaci a zároveň ponechat ostatní součásti beze změny). Reálně je vrstevný model použit například u rodiny protokolů TCP/IP, kde jsou však použity jen čtyři vrstvy.





**Příkladem** (viz obrázek) připomínajícím vrstvý model ISO/OSI může být dopisová komunikace mezi manažery dvou firem (řekněme české a čínské). Jednotlivé vrstvy obou stran spolu zdánlivě komunikují přímo (stejně vrstvy na obou stranách používají stejný protokol, řeč, způsob prezentace dat), ale ve skutečnosti probíhá komunikace od vyšší vrstvy směrem k nejnižší, která jediná disponuje možností přenosu. Na cílové straně dochází naopak k předávání zprávy od nejnižší vrstvy směrem k vyšším.

Každá ze sedmi vrstev vykonává skupinu jasně definovaných funkcí potřebných pro komunikaci. Pro svou činnost využívá služeb své sousední nižší vrstvy. Své služby pak poskytuje sousední vyšší vrstvě.

Podle referenčního modelu není dovoleno vynechávat vrstvy, ale některá vrstva nemusí být aktivní. Takové vrstvě se říká *nulová*, nebo *transparentní*.

Komunikaci mezi systémy tvoří:

1. **komunikace mezi vrstvami jednoho systému**, řídí se pravidly, která se obvykle nazývají *rozhraní* (interface),
2. **komunikace mezi stejnými vrstvami různých systémů**, řídí se *protokoly*.

Na počátku vznikne požadavek některého procesu v aplikační vrstvě. Příslušný podsystém požádá o vytvoření spojení prezentační vrstvou. V rámci aplikační vrstvy je komunikace s protějším systémem řízena aplikačním protokolem. Podsystémy v prezentační vrstvě se dorozumívají prezentačním protokolem. Takto se postupuje stále níže až k fyzické vrstvě, kde se použije pro spojení přenosové prostředí. Současně se při přechodu z vyšší vrstvy k nižší přidávají k uživatelským (aplikačním) datům záhlaví jednotlivých vrstev. Tak dochází k postupnému *zapouzdřování* původní informace. U příjemce se postupně zpracovávají řídicí informace jednotlivých vrstev a vykonávají jejich funkce.

### Fyzická vrstva

Vrstva č. 1, anglicky *physical layer*. **Specifikuje fyzickou komunikaci**. Aktivuje, udržuje a deaktivuje fyzické spoje (např. komutovaný spoj) mezi koncovými systémy. Fyzické spojení může být dvoubodové (sériová linka) nebo mnohabodové (Ethernet).

Fyzická vrstva definuje všechny elektrické a fyzikální vlastnosti zařízení. Obsahuje rozložení pinů, napěťové úrovně a specifikuje vlastnosti kabelů; stanovuje způsob přenosu "jedniček a nul". Huby, opakovače, síťové adaptéry a **hostitelské adaptéry** (Host Bus Adapters používané v síťových úložištích SAN) jsou právě zařízení pracující na této vrstvě.

Hlavní funkce poskytované fyzickou vrstvou jsou:

- Navazování a ukončování spojení s komunikačním médiem.
- Spolupráce na efektivním rozložení všech zdrojů mezi všichni uživatele.
- Modulace neboli konverze digitálních dat na signály používané přenosovým médiem (a zpět) (A/D, D/A převodníky).

### Linková (spojová) vrstva

Vrstva č. 2, anglicky *data link layer*. **Poskytuje spojení mezi dvěma sousedními systémy**. **Uspořádává data** z fyzické vrstvy **do logických celků** známých jako **rámce (frames)**. Seřazuje přenášené rámce, stará se o nastavení parametrů přenosu linky, oznamuje neopravitelné chyby. Formátuje fyzické rámce, **opatřuje je fyzickou adresou** a poskytuje synchronizaci pro fyzickou vrstvu.

Datová vrstva poskytuje funkce k přenosu dat mezi jednotlivými síťovými jednotkami a detekuje případně opravuje chyby vzniklé na fyzické vrstvě. Nejlepším příkladem je Ethernet. Na lokálních sítích založených na IEEE 802 a na některých IEEE 802 sítích jako je FDDI, by tato vrstva měla být rozdělena na **vrstvu řízení přístupu k médiu (Medium Access Control, MAC)** a vrstvu IEEE 802.2 **logické řízení linek (Logical Link Control, LLC)**. Na této vrstvě pracují veškeré **mosty a přepínače**. Poskytuje propojení pouze mezi místně připojenými zařízeními a tak vytváří doménu na druhé vrstvě pro směrové a všesměrové vysílání.

### Síťová vrstva

Vrstva č. 3, anglicky *network layer*. **Tato vrstva se stará o směrování v síti a síťové adresování**. Poskytuje spojení mezi systémy, které spolu přímo nesousedí. Obsahuje funkce, které umožňují překlenout rozdílné vlastnosti technologií v přenosových sítích.

Síťová vrstva poskytuje funkce k zajištění přenosu dat různé délky od zdroje k příjemci skrze jednu případně několik vzájemně propojených sítí při zachování kvality služby, kterou požaduje přenosová vrstva. Síťová vrstva poskytuje směrovací funkce a také reportuje o problémech při doručování dat. Veškeré **směrovače** pracují na této vrstvě

a posílají data do jiných sítí. Zde se již pracuje s hierarchickou strukturou adres. Nejznámější protokol pracující na 3. vrstvě je Internetový Protokol (IP), dalšími jsou ICMP a ARP. Jednotkou informace je paket.

### Transportní vrstva

Vrstva č. 4, anglicky *transport layer*. Tato vrstva zajišťuje přenos dat mezi koncovými uzly. Jejím účelem je poskytnout takovou kvalitu přenosu, jakou požadují vyšší vrstvy. Vrstva nabízí spojově (TCP) a nespojově orientované (UDP) protokoly.

- TCP – Zajišťuje přenos dat se zárukami, který vyžadují aplikace, kde nesmí „chybět ani paket“. Jedná se o přenosy souborů, e-mailů, WWW stránek atd. Záruka se vztahuje na řešení ztrát přenášených paketů, zachování jejich pořadí a odstranění duplikace. Jednotkou posílané informace je na této vrstvě TCP segment.
- UDP – Zajišťuje přenos dat bez záruk, který využívají aplikace, u kterých by bylo na obtíž zdržení (delay) v síti způsobené čekáním na přenos všech paketů a ztráty se dají řešit jiným způsobem (např. snížení kvality, opakování dotazu). Využívá se pro DNS, VoIP, streamované video, internetová rádia, vyhledávání sdílených souborů v rámci sítě DC++, on-line hry atp.

### Relační vrstva

Vrstva č. 5, anglicky *session layer*. Smyslem vrstvy je organizovat a synchronizovat dialog mezi spolupracujícími relačními vrstvami obou systémů a řídit výměnu dat mezi nimi. Umožňuje vytvoření a ukončení relačního spojení, synchronizaci a obnovení spojení, oznamování výjimečných stavů. Do této vrstvy se řadí: NetBIOS, AppleTalk, RPC, SSL. K paketům přiřazuje synchronizační značky, které využije v případě vrácení paket (např. z důvodu, že se během přenosu dat poškodí síť) k poskládání původního pořadí.

### Prezentační vrstva

Vrstva č. 6, anglicky *presentation layer*. Funkcí vrstvy je transformovat data do tvaru, který používají aplikace (šifrování, konvertování, komprimace). Formát dat (datové struktury) se může lišit na obou komunikujících systémech, navíc dochází k transformaci pro účel přenosu dat nižšími vrstvami. Mezi funkce patří např. převod kódů a abeced, modifikace grafického uspořádání, přizpůsobení pořadí bajtů a pod. Vrstva se zabývá jen strukturou dat, ale ne jejich významem, který je znám jen vrstvě aplikační. Příklady protokolů: SMB (Samba).

### Aplikační vrstva

Vrstva č. 7, anglicky *application layer*. Účelem vrstvy je poskytnout aplikacím přístup ke komunikačnímu systému a umožnit tak jejich spolupráci. Do této vrstvy se řadí například tyto služby a protokoly: HTTP, FTP, DNS, DHCP, POP3, SMTP, SSH, Telnet, TFTP.

## FYZICKÁ VRSTVA podrobně.

Fyzické vrstvy se týkají standardy, které definují elektrické, mechanické, funkční a procedurální vlastnosti rozhraní pro připojení různých přenosových prostředků a zařízení (tj. kabelů, modemů apod.) - tedy elektrické parametry přenášených signálů, jejich význam a časový průběh, vzájemné návaznosti řídicích a stavových signálů, zapojení konektorů, a mnoho dalších parametrů technického i procedurálního charakteru. Úkolem entit fyzické vrstvy je pak na základě těchto standardů obsluhovat přenosové prostředky, připojené k příslušným rozhraním, a jejich prostřednictvím zajišťovat přenosy jednotlivých bitů.

### 1. Přenosová média

Patří sem telefonní linky, vícedrátové spoje, dvoudrátová symetrická a nesymetrická vedení, radiové kanály, družicové spoje a optické spoje.

### Typy přenosových médií:

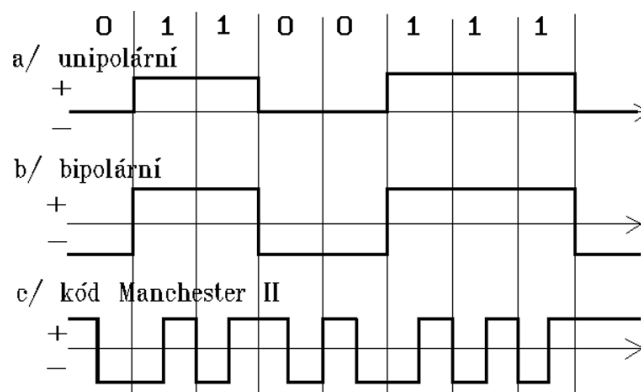
- **Telefonní hovorové kanály** – použití fungující sítě, přenos na značné vzdálenosti, minimální náklady, přenosové pásmo 300 – 3300Hz, analogové dovolily rychlosti od 200 b/s do 52 kb/s
- **Vícdrátové vedení** – přenos na malé vzdálenosti, komunikační kanály s vysokou rychlostí (sběrnice)
- **Radiové kanály** – všesměrové spoje, přenos pro mobilní zařízení (WiFi, Bluetooth); směrové spoje, náhrada metalických spojů, přenos ve veřejných i pronajatých pásmech 2,4;5;10GHz
- **Družicové spoje** – kmitočtová pásma 4/6 a 12/14 GHz
- **Optické trasy** – přenosy dat na velké vzdálenosti (do 10km bez opakování)

### Druhy provedení přenosových médií:

- **Symetrická vedení** – kroucený dvoudrát, levné lokální sítě, typické rychlosti 100 – 1000Mb/s
- **Nesymetrická vedení** – koaxiální kabel, přenosové rychlosti do 10Mb/s

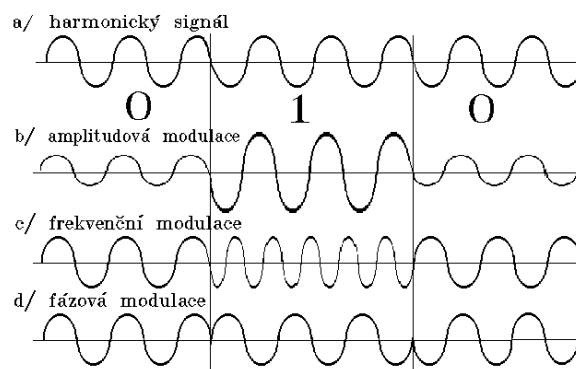
## 2. Kódování a modulace

Neupravený datový signál není vhodný pro přímý přenos komunikačním médiem. Obsahuje stejnosměrnou složku, jejíž přenos lze obtížně zajistit (elektrické vlastnosti kanálu, nutnost galvanického oddělení vedení transformátorem). Signál můžeme zbavit nulové složky vhodným kódováním.



**Přenos kódovaného datového signálu označujeme jako přenos v základním pásmu.** Například síť Ethernet používá kódování NRZ (Manchester II) – v podstatě se jedná o fázovou modulaci.

**MODULACE:** (nosné frekvence)



V sítích se podle doporučení IEEE 802 užívá diferenciální fázová modulace. **Přenos modulovaného datového signálu označujeme jako přenos v přeloženém pásmu.**

## Fyzické rozhraní datového spoje

### Základní pojmy:

modem (modulátor, demodulátor)

ukončující zařízení datového spoje (modem) – DCE  
(Data Circuit Terminating Equipment)

koncové zařízení – DTE (Data Terminal Equipment)

Rozhraní datových spojů jsou definována podle doporučení CCITT a EIA (V.24 a X.21 CCITT)

### Doporučení V.24 CCITT

obdoba EIA RS-232C, nejstarší a nejběžnější rozhraní pro sériový přenos dat, popis mechanických vlastností – konektory DB-25 nebo DB-9

### Doporučení V.28 CCITT

popisuje elektrické vlastnosti, definice napěťových úrovní a jejich přiřazení logickým úrovním, elektrické vlastnosti vysílačů a přijímačů rozhraní

### Doporučení X.21 CCITT

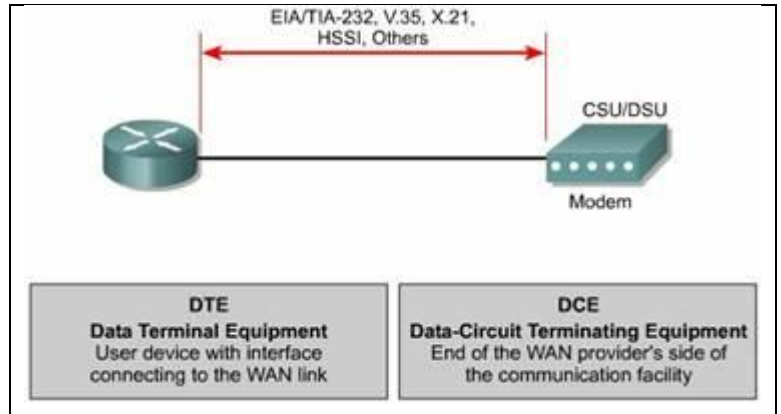
Cílem doporučení je zvětšení povolené délky propojovacího kabelu, snížení počtu obvodů rozhraní. Prakticky se jedná o definici síťového rozhraní sítě s přepojováním kanálů.

### Doporučení V.10/X.26 CCITT

popisuje mechanické vlastnosti – konektory DB-15, nesymetrické vysílače (obdoba EIA RS-423-A)

### Doporučení V.11/X.27 CCITT

symetrické vysílače (obdoba EIA RS-422-A)



## Principy komunikace na fyzickém médiu.

(viz soubor PDF přístupové metody)

**Přístupové metody.** Určují předání dat mezi jednotlivými stanicemi.

**Nedeterministické – (kolizní, náhodné)** využívají metody náhodného přístupu ke sdílenému médiu. Příkladem je Ethernet. Uzly se pokoušejí komunikovat bez jakéhokoli pořadí. Žádný uzel tak nemá garantováno, že se mu podaří přenést určité množství dat za určitou dobu.

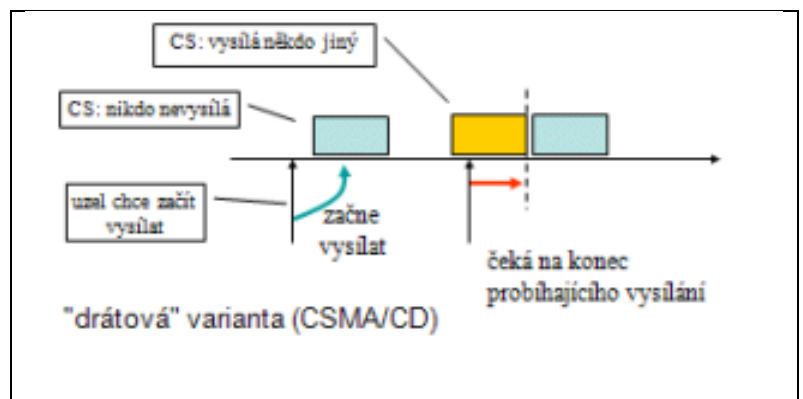
Používají se v sítích, kde jsou přenosy rozesílány všem a každý účastník dostane informace skoro ve stejný okamžik. Pokud uzel chce vysílat, zkontroluje linku, je-li linka obsazená, nebo došlo-li ke kolizi přenos je zrušen, uzel čeká náhodně dlouhou dobu, než zkusí nový přístup znovu

### CSMA/CD (Carrier Sense Multiple Access / Collision Detection)

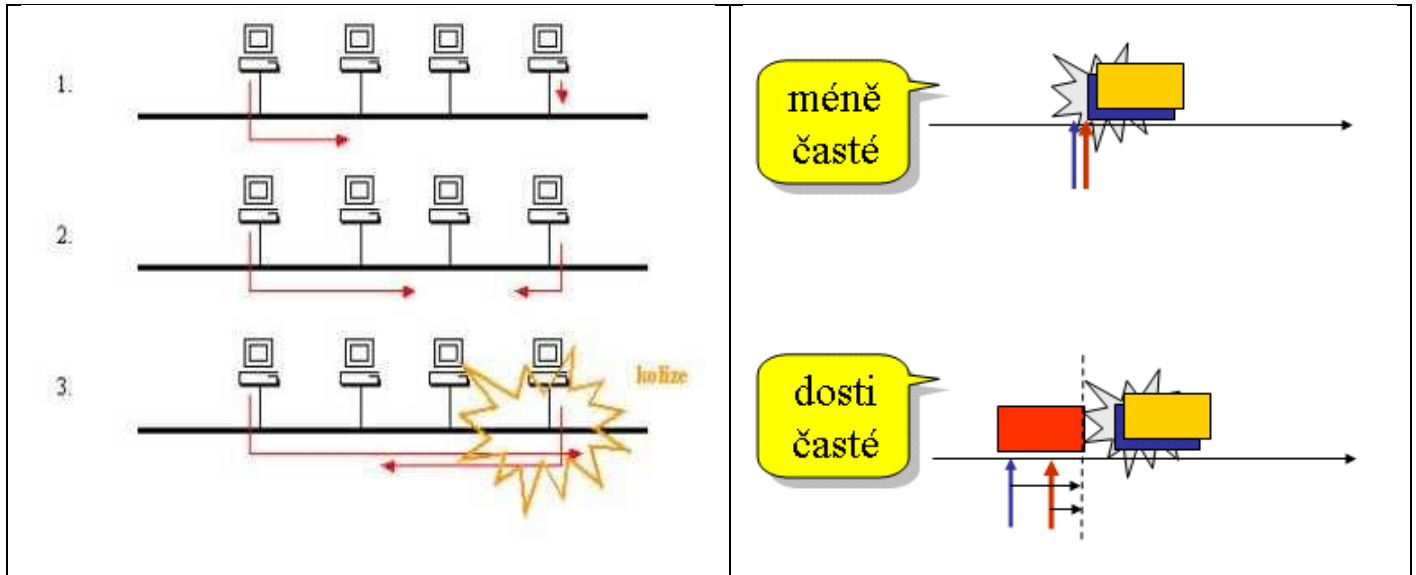
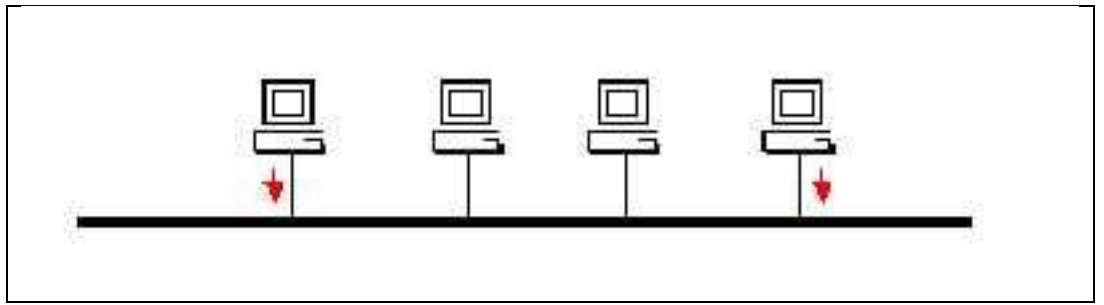
(s nasloucháním nosné a detekcí kolizí) – přístup k síti získá uzel, kterému se jako prvnímu podaří přistoupit k nečinné síti.

#### Princip:

Uzel, který chce vysílat informaci do sítě poslouchá, zda je na síti provoz; pokud je linka obsazena uzel počká náhodně dlouhou dobu, potom provede kontrolu linky, je-li linka volná uzel začne vysílat svůj paket, který se šíří ke všem stanicím zapojeným v síti; uzel pokračuje ve sledování sítě.

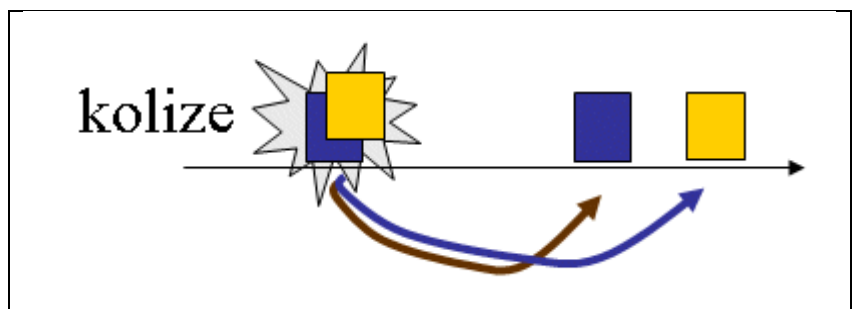


Je možné, že dva či více uzlů začne vysílat ve stejný okamžik => vznik kolize. Kolize je detekována, protože uzly, které vysílaly své pakety, sledují síť a zjistí, že se na přenosovém médiu, vyskytují i jiné informace než tam poslaly.



**Uzel, který detekuje kolizi,** zruší svůj přenos vysláním **JAM** paketu (JAM PATTERN); potom počká náhodně dlouhou dobu a pokusí se k síti přistoupit znovu.

Náhodně dlouhá doba je u každého uzlu jiná; každý uzel naslouchá, a pokud přijde paket, zkontroluje, zda se nejedná o kolizní paket



JAM; pokud je kolizní paket, nepřijme ho; **nejedná-li se o kolizní paket zkontroluje cílovou adresu** a cílová adresa bude pro tento uzel, data přečte. Je-li cílová adresa součástí **broadcastu** – čte data, popřípadě když je cílová adresa součástí **multicastu** – čte data.

Metoda je celkem rychlá. Minimální délka rámce musí být 64 B (některá literatura uvádí délku 72 B). Tato metoda má nejlepší výsledky při menší aktivitě sítě a delších zprávách.

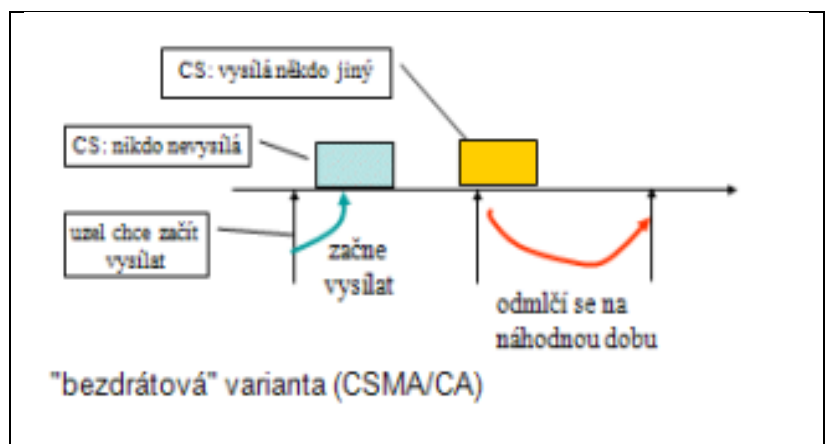
**Příklad sítě: Ethernet, FastEthernet**

### CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance)

(s nasloucháním nosné, předvídání kolize - snaží se vyhnout kolizi) – u této metody je nutné dodržovat **minimální rozestup mezi následujícími pakety** přibližně 200  $\mu$  s.

#### Princip:

Pokud chce uzel vysílat, poslouchá aktivitu na síti; je-li linka obsazená, počká náhodně dlouhou dobu a pokusí se znovu přihlásit. Pokud je síť nečinná vyšle uzel signál **RTS (Request to signal)**, v případě, že je stanice připojena a





tento signál přijme, vyšle potvrzovací signál **CTS** (Clear to send); tyto signály musí být poslány během definovaného časového intervalu; pokud odesílatel obdrží CTS, provede přenos; pokud CTS neobdrží, přenos se odloží; v případě **broadcastu**, RTS je adresováno speciální adresou, nečeká se na CTS a okamžitě jsou data přenášena.

**Příklad sítě: bezdrátový Ethernet (Wifi), síť Apple – Macintosh**

**Deterministické** – (bezkolizní) jsou založeny na řízení přístupu k médium. K řízení je používána **metoda předávání speciálního paketu - peška (token)**. Typickým představitelem technologie používající deterministickou metodu je **Token Ring**.



Velice zjednodušený náhled na funkci Token Ringu lze popsat následovně:

- po síti je přenášén paket nazývaný token.
- uzel, který potřebuje komunikovat, musí počkat, až k němu token dorazí
- může změnit příznak, doplnit hlavičku, naplnit datové pole a odeslat data cílovému uzlu
- ten po obdržení datového paketu zkontroluje kontrolní součty, přijme data, nastaví příslušné příznakové bity a pošle paket opět do sítě
- paket dorazí k tomu uzlu, který data poslal
- tento uzel si prohlédne příznaky a předá je vyšším vrstvám
- vygeneruje prázdný paket (token) a odešle jej
- token je pak předáván mezi uzly na síti až dorazí k uzlu, který má připravena data pro vysílání – a vše se opakuje

Ze znalosti maximální velikosti paketu a počtu uzlů na kruhu lze vypočítat maximální dobu, za kterou se uzlu podaří odeslat příslušné množství dat. Fungování Token Ringu je však o dost složitější než jsme si zde v několika větách popsali (např. někdo musí hlídat, zda se na síti vyskytuje token). Technologie je tedy logicky složitější a tím i dražší než Ethernet a jsou dva zásadní důvody, proč Token Ring nedoznal takového rozšíření a dokonce byl zastaven jeho další rozvoj. Mezi deterministické metody patří i tzv. **Pooling**, kdy řídicí počítač určuje, která stanice bude vysílat. U této metody musí být spolehlivý řídicí počítač.

**Příklad sítě: Cambridge Ring, Token Ring, FDDI**

## **LOKÁLNÍ SÍŤ - úvod.**

Definice: Lokální síť (LAN) – **geograficky omezená síť**, propojení velkého množství počítačů na vzdálenost stovek metrů, odklon od klasického přepojování, využívají mnohonásobného přístupu ke sdílenému médiumu.

**Vývojové typy LAN:**

- **ALOHA** – experimentální radiová síť, metoda náhodného přístupu ALOHA
- **ETHERNET** – experimentální síť firmy XEROX, sdílené médium je koaxiální kabel, rychlost do 10Mb/s, používá se dodnes poměrně složitá přístupová metoda CSMA/CD (Carrier Sense Multiply Access / Collision Detection)
- **Cambridge Ring** – univerzitní síť několika počítačů, přenosové médium je levný dvojitý kroucený dvoudrát, přenosová rychlost do 10Mb/s, díky uzavřené topologii - jednoduché řízení



## Standard IEEE 802

**IEEE 802** je rodina IEEE standardů pojednávajících o sítích LAN a MAN.

- IEEE 802.1 Bridge and Network Management
  - IEEE 802.1AE - MAC Security
  - IEEE 802.1Q - VLAN
  - IEEE 802.1X - Network Access Control
  - IEEE 802.1aq - Shortest Path Bridging (SPB)
- IEEE 802.2 - LLC - Logical Link Control
- **IEEE 802.3 - Ethernet**
- IEEE 802.4 - Token Bus
- IEEE 802.5 - Token Ring
- **IEEE 802.11 - Wi-Fi**
- IEEE 802.12 demand priority access method
- IEEE 802.13 Kabeláž kat.6 - 10Gb lan
- IEEE 802.15 Wireless PAN
  - IEEE 802.15.1 Bluetooth
  - IEEE 802.15.4 ZigBee
- IEEE 802.16 Broadband Wireless Access WiMAX
  - IEEE 802.16e (Mobile) Broadband Wireless Access
- IEEE 802.17 Resilient packet ring
- IEEE 802.18 Radio Regulatory TAG
- IEEE 802.19 Coexistence TAG
- IEEE 802.20 Mobile Broadband Wireless Access
- IEEE 802.21 Media Independent Handoff
- IEEE 802.22 Wireless Regional Area Network

### IEEE 802.3 - SÍŤĚ ETHERNET

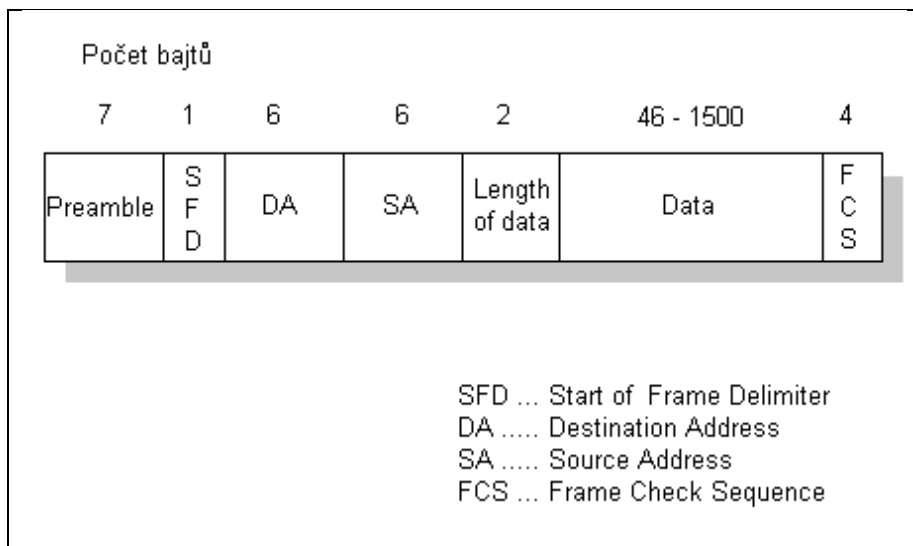
Původní standard **IEEE 802.3** z roku 1985 používal pouze **koaxiální kabely**, nejprve jen známý "**tlustý**", nebo také podle jeho barvy "žlutý" kabel (**Thick Ethernet**). Segment mohl být **dlouhý až 500 m**, více segmentů mohlo být propojeno opakovači (repeater) až do **max. vzdálenosti 2500 m** mezi **nejvzdálenějšími uzly, kterých mohlo být max. 1024**. Síťová karta uzlu byla k segmentu připojena speciálním transceiverem a přípojným kabelem.

Celá logika protokolu Ethernetu je implementována v obvodech síťového adaptéru. Protokolový paket má pevný formát, s datovým polem v rozmezí 46 - 1500 bajtů. (viz obrázek)

Později byla do standardu doplněna podpora "**tenkého**" koaxiálního kabelu (**Thin Ethernet**) a **ještě později** **nestíněného krouceného dvou páru (UTP - unshielded twisted pair)**.

Standardní Ethernet tak vydržel spokojeným uživatelům 15 let, než v roce 1995 byla organizací IEEE přijata specifikace **802.3u Fast Ethernet**. Rychlost se zvýšila skokově - na **100 Mb/s**, což bylo zvýšení na desetinásobek!

V té době byla jediná **možnost, jak**



zvýšit propustnost páteře nebo připojení serverů - použít technologii FDDI

**FDDI** (Fiber Distributed Data Interface). Technologie FDDI byla v první polovině 90. let prakticky jedinou možností, jak vybudovat rychlou síť. Přitom již v té době šlo o osvědčenou a vysoce spolehlivou technologii s dokonalým managementem, s přenosovou rychlostí 100 Mb/s pro zatížená síťová centra, propojování vzdálených budov a areálů a budování metropolitních sítí (MAN - Metropolitan Area Network).

Vzhledem k její značně vysoké ceně si toto řešení mohl ale jen málokdo dovolit. Proto byl Fast Ethernet velmi rychle a dobře přijat širokou uživatelskou veřejností.

Zvýšením rychlosti na desetinásobek má ale za následek zmenšení maximální vzdálenosti mezi uzly (velikosti kolizní domény) na desetinu. Vyplývá to z principu použité přístupové metody Ethernetu, metody CSMA/CD.

Ve standardu **Fast Ethernet** se zmenšila vzdálenost propojení opakovačů a přepínačů na 100 m v případě krouceného dvou páru a 412 m v případě optických kabelů. Tento problém se řeší používáním topologie zhroucené (soustředěné) páteře, angl. collapsed backbone. Překonání vzdálenosti větší než 412 m umožnily přepínače s porty schopnými plně duplexního provozu, kde se délka propojovacích optických kabelů zvětšila na 2000 m.

Ethernet se tak stal velmi flexibilním, škálovatelným médiem, umožňujícím soužití sdílených segmentů spolu s přepínanými, dedikovanými spoji, a to 10 Mb/s i 100 Mb/s segmentů s různou kabeláží.

Při implementaci Fast Ethernetu nastal typický problém páteřních sítí, na který postupně narazí všichni návrháři a administrátoři distribuovaných sítí. Použijete-li vyšší přenosovou rychlost na sdílenou páteř, postupně např. se zlevňováním technologií spolu se zvyšujícími se požadavky uživatelů, dochází k pronikání rychlejších technologií "dolů" směrem k uživatelům. Logicky pak přestává páteř svojí přenosovou kapacitou stačit zvyšujícím se nárokům. A toto se záhy stalo i v případě 100 Mb/s Ethernetu.

Tak vznikl brzy tlak na další zrychlení a opět desetkrát rychleji - 1000 Mb/s.

Dnes je i tato gigabitová rychlost již skutečností a ve výhledu je zvýšení přenosové rychlosti páteře na 10 Gb/s.

### Používaná komunikační média:

Tabulka ukazuje vývoj standardu Ethernet, přenosové rychlosti, typu kabeláže a max. délky jednotlivých segmentů. Zavedená terminologie standardů používá notaci "nBase-X", kde n je nominální přenosová rychlost v Mb/s, Base - jedná se o přenos v základním pásmu a X je typ média kabelu.

Typ Ethernetu	Rychlost	Max. délka segmentu
10Base-5	10 Mb/s "tlustý" coax	500 m
10Base-2	10 Mb/s "tenký" coax	185 m
10Base-T (Twist pair)	10 Mb/s UTP	100 m
10Base-FL (Fiber)	10 Mb/s optika	2000 m *
100Base-TX (X - Fast)	100 Mb/s UTP (2 páry pro přenos)	100 m
100Base-T4 (4 - páry)	100 Mb/s UTP (4 páry pro přenos)	100 m
100Base-FX	100 Mb/s optika	412 m HDX *, 2000 m FDX *

\* při použití mnohavidového vlákna 62,5 mikronů, HDX - half duplex, FDX - full duplex

## Jednotlivé typy a označení komunikačních médií:

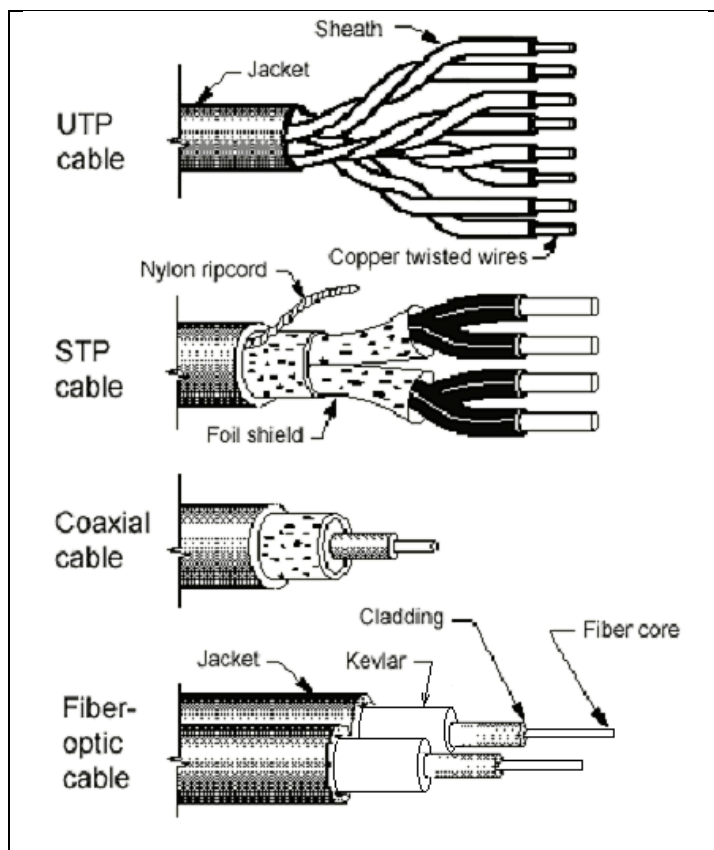
**UTP kabel (Unshielded Twist Pair)** kroucená dvojlinka, označení (10) 100BASE-(T), -TX, -T4  
(T – twist pair (kroucený dvoudrát) X – Fast Ethernet, 4 – zapojené všechny páry)

**STP kabel (Shielded Twist Pair)**, označení stejné

**Tenký koaxiální kabel (Thin Ethernet)** – lokální rozvody, označení **10BASE 2** (10 – komunikační rychlost v Mb/s, BASE – přenos probíhá v základním pásmu bez modulace, 2 – maximální délka kabelu mezi porty ve stovkách metrů)

**Tlustý koaxiální kabel (Thick Ethernet)** – páteřní rozvody, označení **10BASE 5**

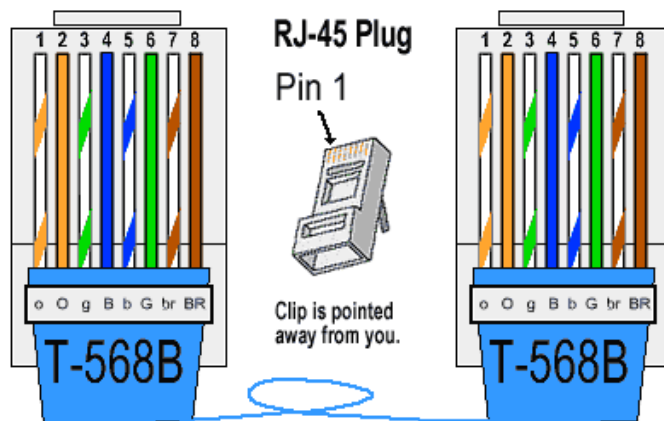
**Optický kabel – FX, SX, LX**



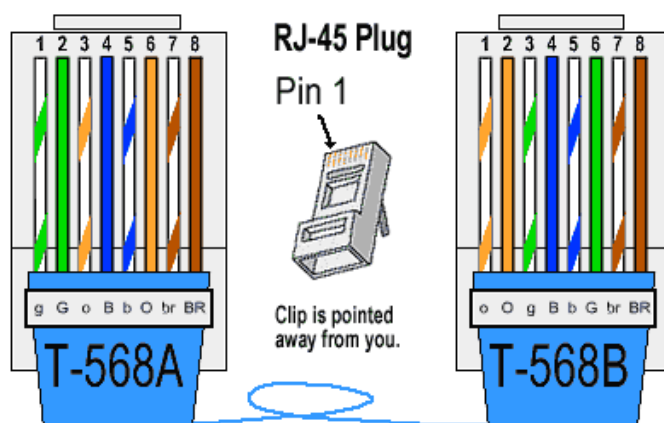
### Typy zapojení kabelů UTP (STP):

Pin	T568A č.páru	T568B č.páru	T568A barva	T568B barva	piny v pohledu na konektor (zásuvka je obráceně)
1	3	2	bílý/zelený	bílý/oranžový	
2	3	2	zelený	oranžový	
3	2	3	bílý/oranžový	bílý/zelený	
4	1	1	modrý	modrý	
5	1	1	bílý/modrý	bílý/modrý	
6	2	3	oranžový	zelený	
7	4	4	bílý/hnědý	bílý/hnědý	
8	4	4	hnědý	hnědý	

## Rovný (Straight) kabel



## Křížový (Cross) kabel



## Speciální kabely:

### Console kabel a konektorové piny

Použijeme RJ-45-na-RJ-45 rollover kabel a RJ-45-na-DB-9 female DTE adaptér (označený TERMINAL) k připojení console portu do PC, s běžícím emulačním terminálovým softwarem (Hyperterminal). Tabulka zobrazuje signály a piny asynchronního sériového console portu, RJ-45-na-RJ-45 rollover kabelu, a RJ-45-na-DB-9 female DTE adaptéru (označeného TERMINAL).

Tabulka signálů Console portu, kabelů a DB-9 adaptéru				
Console port (DTE)	RJ-45-na-RJ-45 Rollover kabel		RJ-45-na-DB-9 Terminal adaptér	PC konzole
Signál	RJ-45 Pin	RJ-45 Pin	DB-9 Pin	Signál
RTS	1 <sup>1)</sup>	8	8	CTS
DTR	2	7	6	DSR
TxD	3	6	2	RxD
GND	4	5	5	GND
GND	5	4	5	GND
RxD	6	3	3	TxD
DSR	7	2	4	DTR
CTS	8 <sup>1)</sup>	1	7	RTS

<sup>1)</sup> Pin 1 je připojen uvnitř na pin 8.

### Auxiliary kabel a konektorové piny

Použijeme RJ-45-na-RJ-45 rollover kabel a RJ-45-na-DB-25 male DCE adaptér (označený MODEM) k připojení auxiliary portu na modem. Tabulka zobrazuje signály a piny asynchronního sériového auxiliary portu, RJ-45-RJ-45 rollover kabelu, a RJ-45-DB-25 male DCE adaptéru (označeného MODEM).

Tabulka signálů Auxiliary portu, kabelů a DB-25 adaptéru				
AUX Port (DTE)	RJ-45-na-RJ-45 Rollover Cable		RJ-45-na-DB-25 Modem Adapter	Modem (DCE)
Signál	RJ-45 Pin	RJ-45 Pin	DB-25 Pin	Signál
RTS	1	8	4	RTS
DTR	2	7	20	DTR
TxD	3	6	3	TxD
GND	4	5	7	GND
GND	5	4	7	GND
RxD	6	3	2	RxD
DSR	7	2	8	DCD
CTS	8	1	5	CTS

## Optické vlákno – optický kabel

Světlo použité v optických vláknech je jeden typ elektromagnetické energie. Důležitá vlastnost každé energetické vlny je vlnová délka. Elektromagnetické vlny se šíří rychlostí 300 000 km/s ve vakuu. V optických vláknech se používá světlo o vlnové délce 850 nm, 1310 nm, nebo 1550 nm. Tyto délky byly vybrány, protože se šíří lépe než ostatní vlnové délky. V optických vláknech je vysoká bezpečnost přenášených dat, protože signál je jen uvnitř vlákna a nedá se odposlouchávat. Na přenos se používají vždy 2 vlákna, jedno na příjem a druhé na vysílání. Jeden kabel může obsahovat 2 – 48 oddělených vláken i více.

Základní pojmy z optiky:

**Odraz světla:** světelný paprsek prochází z jednoho okraje materiálu na druhý, část světelné energie bude odražena zpět

**Lom světla:** zakřivení světelného paprsku při průchodu dvěma látkami je důvod, proč může světelný paprsek cestovat v optickém vlákne také v zatáčkách.

**Optická hustota materiálu:** popisuje, jak se množství světelných paprsků zmenší při průchodu materiálem, nejlepší je ve vakuu.

**Index lomu:** poměr rychlosti světla v materiálu k rychlosti světla ve vakuu, čím je menší, tím je materiál vhodnější.

### Odraz

Úhel, pod kterým světelný paprsek dopadá na hladký, lesklý, povrch je stejný, jako úhel paprsku, pod kterým se odráží.

### Lom

Při dopadu světelného paprsku na rozhraní 2 transparentních materiálů, dochází k rozdělení paprsku na 2 části. Část první se odrazí v původním materiálu podle úhlu dopadu, a druhá prochází do druhého prostředí. Když je úhel 90 stupňů, paprsek prochází přímo sklem, když ne potom se láme. Když paprsek prochází z prostředí s menším indexem lomu do prostředí s větším, potom se paprsek láme ke kolmici k povrchu v místě dopadu a opačně.

### Úplný optický odraz

#### Podmínky pro optická vlákna:

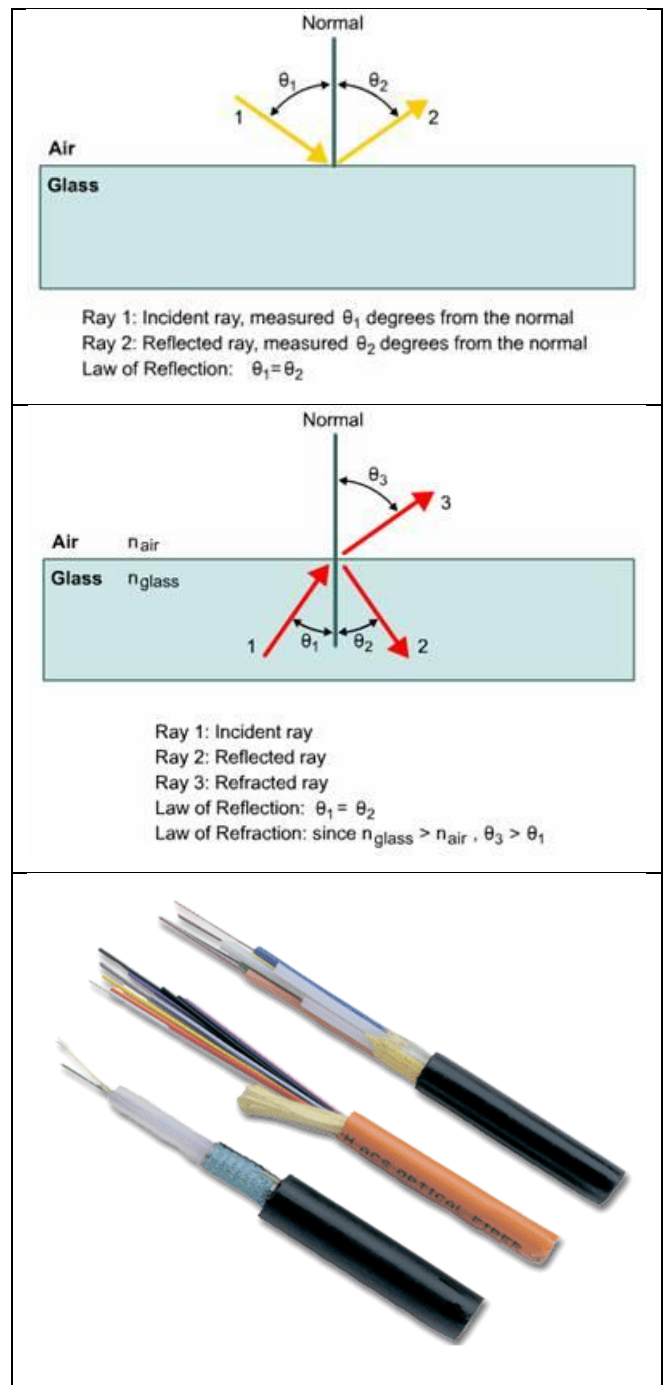
1. jádro optického vlákna by mělo mít větší index lomu, než plášť, který ho obklopuje
2. úhel dopadu světelného paprsku by měl být větší jak kritický úhel pro jádro a obal

**Úplný vnitřní odraz:** dosáhneme jej při splnění obou podmínek, vlastní světelný paprsek bude úplně odražený zpět do vnitřku vlákna.

#### Typy optických vláken:

**Vlákno multimodové (vícevidové) –** průměr vlákna 50 – 62,5 mikrometru, umožňuje použití více paprsků, zdrojem světla je většinou LED dioda

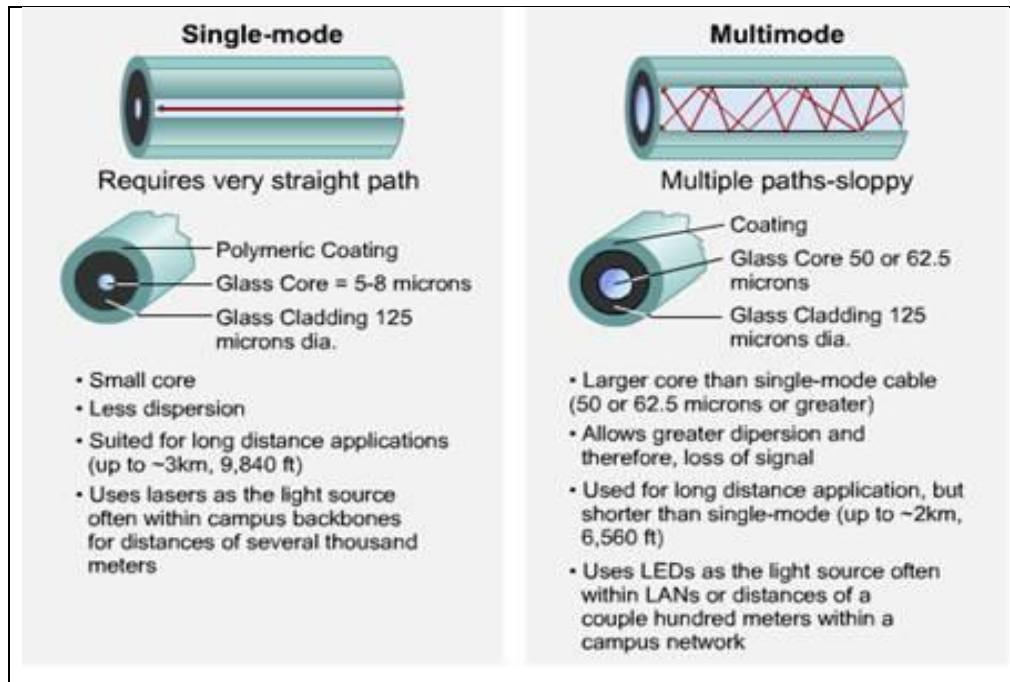
- označení **100BASE-FX** – dvě multimodová vlákna
- označení **1000BASE-SX MM** nebo **LX MM** - délka do 2 km



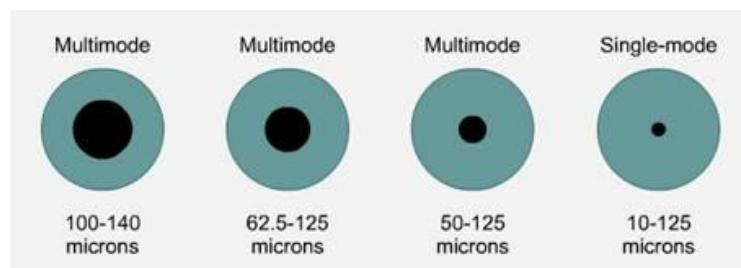
**Vlákno singlemodové (jednovidové)** -- průměr vlákna 8 – 10 mikrometrů, používá se pouze jeden paprsek, zdrojem světla je laser

- označení **1000BASE-LX SM** - délka až 5 km

**Šíření paprsků v optickém kabelu:**

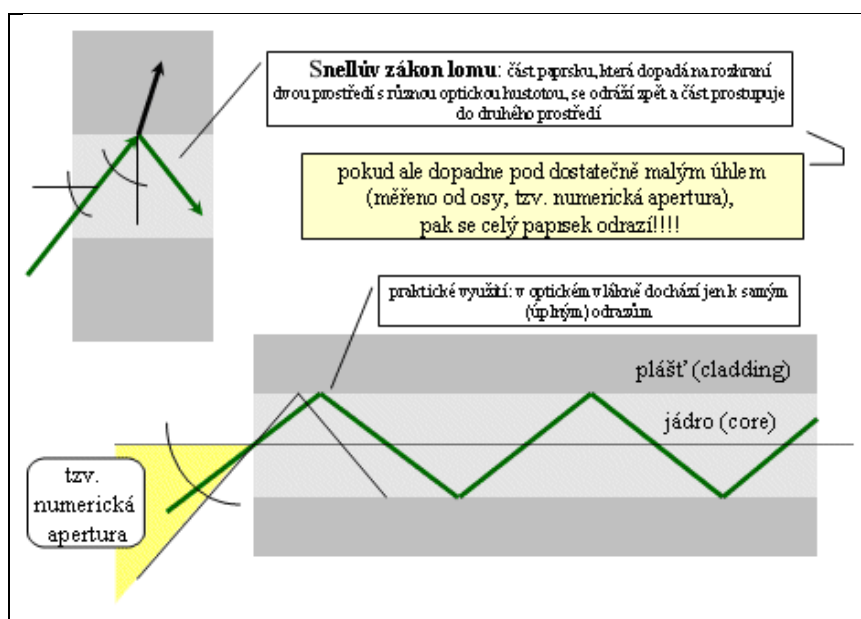


**Porovnání rozměrů vláken:**



## Numerická apertura

Je úhel, pod kterým může paprsek do vlákna dopadat, aby došlo k úplnému odrazu





## Komponenty optické trasy.

Na přenos dat v síti se používají elektrické signály, proto je potřebné v optických přenosových zařízeních použít přijímače a vysílače na převod elektrického signálu do a z optiky zpět.

### Vysílač (Transmitter):

Používá 2 typy zdroje optického signálu : LED, LASER

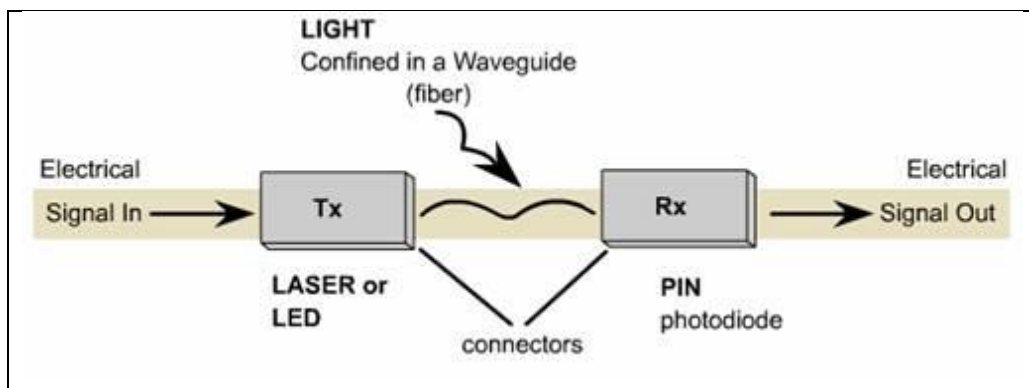
### Přijímač (Receiver)

jeho úlohou je přijmout signál a zpracovat ho na elektrické signály. Používají se



fotodiody PIN. PIN diody jsou

vyráběné na vlnovou délku

850, 1310 nebo 1550nm. Kabely jsou ukončené konektory.



V tabulce je **přehled používaných konektorů**.

Konektor typu SC – na ústupu	Konektor typu ST – nejhorší, bajonet kroutí vlákno
	
Konektor LC - moderní	Konektor E2000 – nejpreciznější, nejdražší, telecom.
	

## Signály a šum v optických médiích

Optické sítě jsou vhodné pro MAN a WAN sítě. Velmi podstatným faktorem jsou **ztráty**. Jsou způsobené mikroskopickými nečistotami ve vlákne, ty způsobují vytváření malého množství tepla. Dále jsou to **různé průměry vlákna a nerovnosti montáže v konektorech**. **Měří se v dB**, a znamená, kolik procent energie bylo ztraceno. Testování je velmi důležité a záznamy musí být zachovány. **Nejpoužívanější měřicí přístroje jsou Optický ztrátový měřič a Optický časový doménový reflektometr**.

## Instalace, ošetření, a testování optických kabelů

Hlavní příčiny útlumu (ztrát):

1. nesprávná instalace, přílišný tah a ohýbání
2. absorpce vlákna
3. nehomogenita vlákna
4. rozptyl ve vlákne

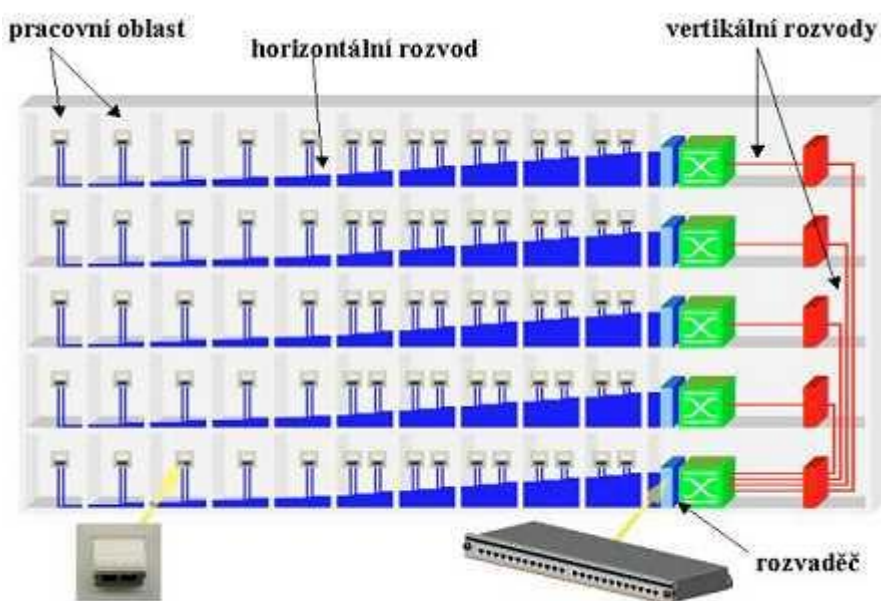
## Výstavba lokálních sítí formou strukturované kabeláže.

Strukturovaná kabeláž tvoří základní prvek infrastruktury moderních lokálních počítačových sítí. Kabelový systém umožňuje nejenom přenos dat, ale je používán také na propojení telefonů, zejména pak v nových budovách nebo v případě rekonstrukce starých telefonních rozvodů. Prostřednictvím přizpůsobovacích prvků (BALUN) a převodníků lze strukturovanou kabeláž používat i pro jiné komunikační systémy – např. pro přenos videosignálu (rozhraní HDMI). Strukturované kabeláže jsou budovány na základě doporučení a norem.

### Doporučení TIA/EIA-586-A.

To dělí kabelový systém na **dvě základní lokality – rozvaděč** (wiring closet) a **pracovní oblast** (working area). Rozvaděč a pracovní oblast jsou **spojené horizontálním rozvodem**. Vyskytuje-li se v místě více rozvaděčů, jsou rozděleny na **centrální rozvaděč - MDF** (main distribution facility) a **podružné rozvaděče - IDF** (intermediate distribution facility). Centrální rozvaděč MDF se v síti vyskytuje pouze jeden a jde o nejvýznamnější komponentu kabeláže neboť zde jsou většinou soustředěny centrální prvky jako jsou směrovací přepínače, připojení k Internetu, telefonní ústředna, ... Podružných rozvaděčů IDF může být v kabelážním systému blíže nespecifikované množství. Platí však pravidlo, že každé patro by mělo mít vlastní rozvaděč, navíc minimálně jeden na každých 1000 m<sup>2</sup>.

**Propojení MDF a IDF je realizováno prostřednictvím tzv. vertikálních (páteřních) rozvodů**. Většinou



bývají optické, i když mohou být i metalické. **Páteřní kabely jsou nejen v rámci budovy, ale i mezi budovami**. V tom případě jsou pro datové komunikace používány výhradně kabely optické.

**V MDF a IDF jsou umístěny propojovací systémy, často nazývané propojovací panely (patch panel)**, připevněné na zdi nebo **v 19" rozváděčových skříních (RACK)**. Zadní část propojovacích systémů slouží pro ukončení kabelů horizontálních rozvodů – na obrázku jsou označeny modře. Přední část propojovacích

systémů je osazena buď **konektory RJ45** pro snadné propojení s aktivními prvky (označeny zeleně) **nebo speciálními zářezovými konektory závislými na výrobci** – používají se pro možnost rozebrání kabelu na jednotlivé

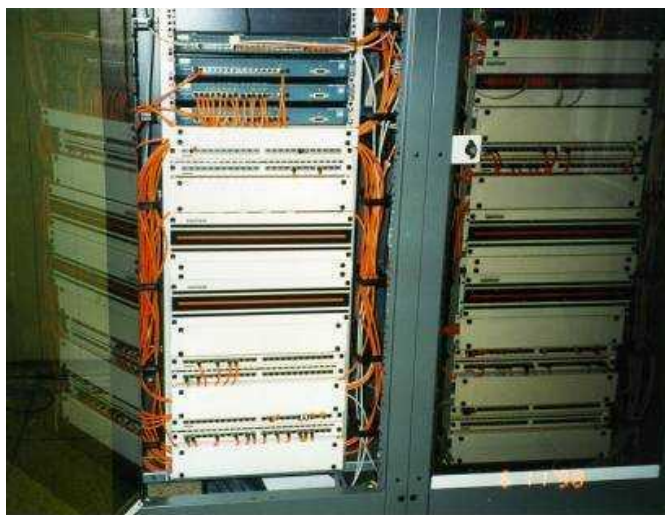
páry (Alcatel IDC, Lucent typ 110, ...). Dříve se vyskytovaly i přípojky realizované tzv. Telco kabely. Ty však byly vhodné pro pomalejší přenosové technologie a v moderních kabelových systémech již prakticky nemají místo.

Horizontální rozvody většinou bývají metalické, i když mohou být i optické. Jak již bylo řečeno, jeden jejich konec je na propojovacím panelu, na druhý je ukončen v zásuvce. Zásuvky nabízejí prostřednictvím konektoru RJ45 možnost připojit prakticky libovolný koncový prvek (telefon, tiskárnu, počítač, ...).

Pro připojování koncových prvků k zásuvce a propojování portů aktivních prvků s propojovacím panelem se používají propojovací kabely (patch cable nebo patch cord).

Jak propojovací panely, tak i zásuvky mají možnost popisu. Ten by měl být v rámci jednoho kabelu na obou koncích totožný. Systém popisování je většinou spojen s číslováním místností v rámci budovy a s pořadím zásuvky. Spojit číslování přípojných míst s číslem místnosti je nejvhodnější, neboť v mnoha případech dochází, ve snaze o úspory, k poddimenzování počtu přípojných míst a tím i k pozdějšímu dodělávání nových přípojných míst. Narušení posloupnosti čísel zásuvek je pak zbytečně matoucí.

Na obrázku je vidět osazená 19" skříň (RACK)



s aktivními prvky, propojovacími panely a poměrně důležitou součástí skříní, na kterou se leckdy zapomíná. Jsou jí vodičí lišty pro propojovací kabely, nejen horizontální (skryta za plechy se 4mi rychloupínači), ale i vertikální (dobře viditelné po okrajích panelů, držící oranžové propojovací kabely).

Součástí skříně může poměrně široká škála doplňků. Tím nejběžnějším je ventilační jednotka s termostatem řízenými ventilátory. Jedním z nejneobvyklejších je hlídací systém, vybavený sirénou, majákem a pagerem schopným ztřípít poplach v případě, že neoprávněná osoba otevře skříň.

Kabely horizontálních i vertikálních rozvodů se umísťují do žlabů (plastových nebo kovových), trubek, podhledů, vodičí můstky, ... Žlabů existuje poměrně široká škála. V rozsahu od jednoduchých se zásuvkami montovaných na zdi poblíž nich až po ozdobné žlaby se zapuštěnými zásuvkami. Ozdobné žlaby jsou z plastu, plechu nebo z eloxovaného hliníku s výběrem barvy a mnoha doplňky parapetního systému. Ve složitějších (komorových) žlabech lze dokonce vést napájecí kabely s datovými rozvody v souběhu (zejména pak, jde-li o stíněné kabelové systémy).

#### **Kategorie rozvodu:**

- KAT 3 – zapojené pouze dva páry vodičů, rychlost do 10 Mb/s (nepoužívá se)
- KAT 5 – zapojené všechny vodiče, pro přenos pouze dva páry, ostatní nevyužité, rychlost do 100 Mb/s
- KAT 6 – zapojené všechny vodiče, pro přenos všechny páry, rychlost do 1 Gb/s

#### **Pravidlo při instalaci UTP(STP) kabelu:**

Pozor: Vzdálenost mezi koncem jednotlivých rozpletených vodičových párů a koncem izolace kabelu resp. rozpletení je maximálně 12,5 mm

#### **Připojovací systémy strukturované kabeláže.**

Vnitřní připojení UTP kabelu ke konektorům, používají se dva zářezové systémy:

- KRONE – starší systém vycházející z technologie používané u telefonní kabeláže
- S110 – novější systém, dnes nejčastěji používaný

Tyto systémy jsou doplněné mechanickou fixací přívodního kabelu, problémy s fixací byly důvodem vzniku dalších konektorových systémů (např. PanJack firmy Panduit)

## LINKOVÁ VRSTVA - podrobně

**Hlavní úkol** – bezchybný přenos mezi **sousedními uzly**. Používá **fyzické adresování** – Media Access Control – odtud název adresa MAC.

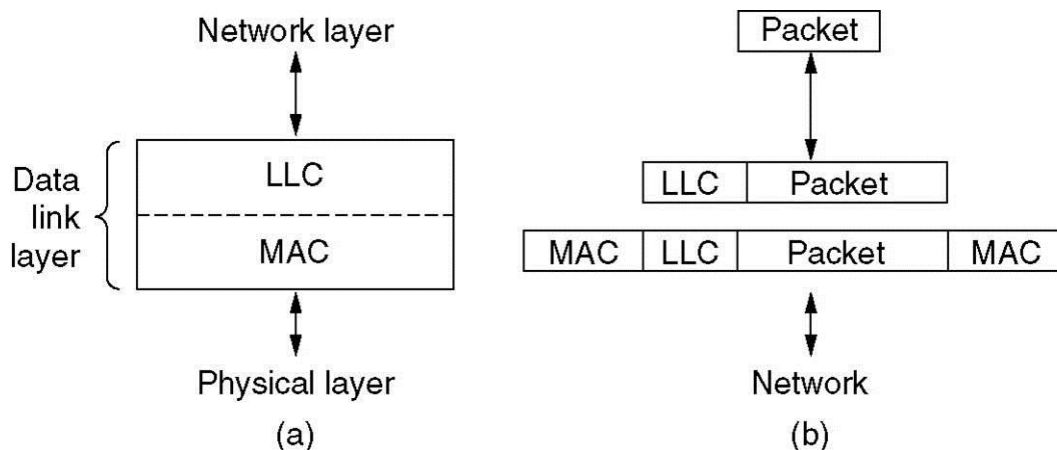
**Adresa MAC** – 48 bitů, reprezentovaných 12 hexadecimálními číslicemi, **rozdělena na dvě části**. Prvních 6 číslic je podle číselníku IEEE identifikační kód výrobce případně podle číselníku OUI kód dodavatele, zbývajících 6 číslic je sériové číslo portu nebo hodnota určená výrobcem. Adresa je uložena (vypálena) v paměti ROM (odtud někdy název BIOS). Při inicializaci portu (adaptéru) se adresa MAC zkopíruje do paměti RAM prvku (adaptéru), síťový adaptér používá MAC adresu k posouzení, zda se přijatá data mají přímo předat vyšším vrstvám protokolu ke zpracování **bez účasti procesoru**.

Aktivní prvek pracující na linkové vrstvě použije MAC adresu k určení přímého souseda a tím i portu, na který budou přijatá data poslána.

Př.: V přijatých datech je cílová adresa MAC : FFFF FFFF FFFF

=> data projdou bez omezení celou sítí a v síťových adaptérech budou přímo předána vyšším vrstvám ke zpracování bez účasti procesoru.

Linková vrstva může být rozdělena na dvě podvrstvy. Jednou z nich je podvrstva MAC a druhou je vyšší podvrstva LLC (Logical Link Control) např. v systému FDDI.



### Funkce MAC podvrstvy:

- zajišťuje fyzické adresování,
- je hardwarově závislá
- zajišťuje řízení přístupu k médiu
  - **deterministické** – lze určit maximální časový interval, ve kterém se pracovní stanice dostane k médiu
    - **Token passing** (předávání peška) – může se ztratit pešek nebo se chybně objeví další, proto je zde Monitor, který předávání sleduje a řídí. Monitor je zvolen dohodou stanic (např. vypne-li se aktivní). Implementace je složitá (software).
    - **Polling** (řízený přístup) – řídící počítač určuje, která stanice bude vysílat. Musí být spolehlivý řídící počítač.
  - **nedeterministické - stochastické - náhodné** (contention = soupeření) – nelze zaručit časový interval, takže stanice musí čekat
    - **CSMA/CD** (Carrier Sense Multiple Access/Collision Detection)
    - **CSMA/CA** (Carrier Sense Multiple Access/Collision Avoidance)

### Funkce LLC podvrstvy:

- zajišťuje multiplexaci protokolů vysílaných na MAC vrstvu (vysílání) a demultiplexaci (příjem)
- není hardwarově závislá
- realizuje řízení toku a zabezpečení proti chybám

**Řízení toku** pomocí tzv. ARQ (Automatic Repeat-request) metodami:

- Jednotlivé potvrzování (Stop&Wait)

- Kontinuální potvrzování s návratem (Go-Back-N)
- Kontinuální potvrzování se selektivním opakováním (Selective Repeat)

### Detekce a korekce chyb

Realizace např. pomocí:

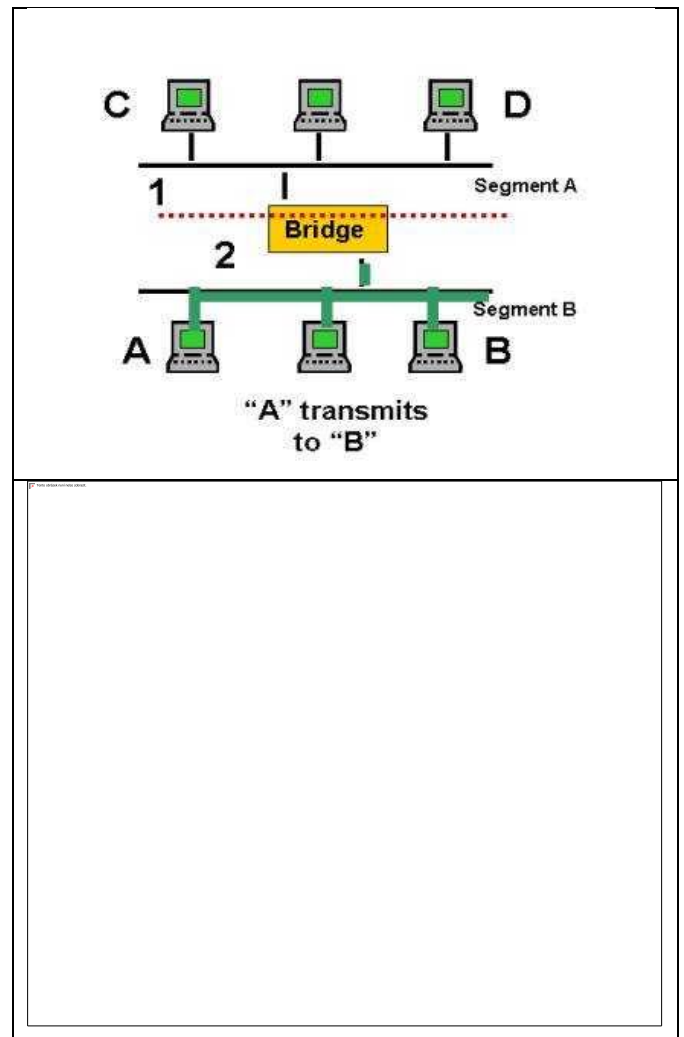
- parita (jednoduchá lichá/sudá, křížová, ...)
- kontrolní součet (CRC)
- Hammingovy kódy (samoopravné kódy)

## Aktivní síťové prvky vrstvy:

**Můstek (bridge)** – dvouportové zařízení které odděluje provoz dvou segmentů sítě na základě učení se fyzických (MAC) adres uzlů na obou portech, na základě těchto adres můstek buď data na druhou stranu propouští, nebo nepropouští; můstek pracuje na druhé vrstvě modelu OSI (linková vrstva) a proto je protokolově nezávislý, je však závislý na používané síťové technologii (přenosové metodě); můstek odděluje kolizní domény, ale rozšiřuje broadcastovou doménu; filtrační schopnost platí s jedním omezením – vztahuje se pouze na Unicast pakety. NonUnicast pakety (Multicast, Broadcast) jsou propouštěny.



**Přepínač (switch)** – vysokorychlostní multiportový můstek, který přináší nové významné vlastnosti:

1. umožňuje paralelní komunikaci mezi různými porty (tzn. např. dvojice portů 2-3, 5-9, 6-4, ... mohou komunikovat současně);
2. umožňuje aplikaci vysokorychlostních portů a pomocí inteligentního používání vyrovnávacích pamětí rozdělit provoz vysokorychlostního portu do několika portů s nižší rychlostí/;
3. vedle standardního polovičně duplexního provozu přináší teoreticky dvakrát rychlejší plně duplexní provoz;
4. přepínač odděluje kolizní domény, ale rozšiřuje broadcastovou doménu (v případě nonunicastového paketu se chová jako rozbočovač – tj. pošle tento paket na všechny porty).





Příklady přepínačů:

<b>Jednoduchý koncový (SOHO) přepínač</b> Laciné řešení, plug and play bez možnosti řízení	<b>Páteřní říditelný přepínač</b> Pokročilé možnosti konfigurace, VLAN, bezpečnostní mechanismy, dálková správa,..., podstatně dražší
	

### Smyčky a mechanismus STP

Na základě obrázku si jistě dokážete představit, že teoreticky stačí jeden NonUnicast k tomu aby zahltil síť. Například přijde na některý z přípojných portů přepínače P1.

Ten jej pošle na všechny ostatní porty včetně těch, na něž jsou připojeny P2 i P3. P2 jej pošle na všechny porty mimo toho, na kterém jej přijal. Tím se paket dostává na P3 – ten jej posílá na P1,

odtud jde na P2 a zase na P3, atd. Nekonečné kolečko je hotové. Původní paket od stanice připojené na P1 je ovšem šířen i druhou stranou, tj. z P1 na P3, z P3 na P2 a z P2 na P1, atd.

Jedinou cestou na druhé vrstvě OSI jak těmto nekonečným přeposíláním paketů zabránit je zabránit vytváření smyček. Toho se dá dosáhnout pečlivým návrhem, realizací a rozvojem sítě nebo **automatizovaným mechanismem nazývaným Spanning Tree Protocol (STP)**. Můstky a přepínače tento protokol používají.

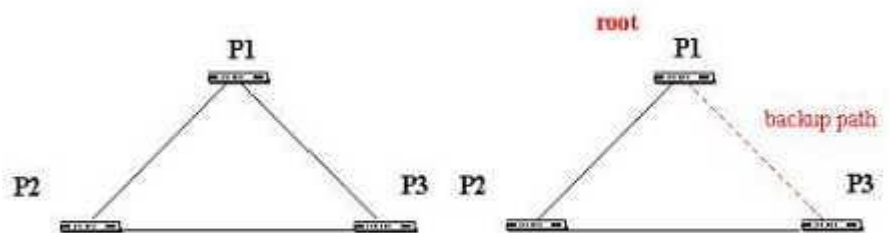
Hlavní význam STP je v tom, že uzavře redundantní cesty, ale zároveň umožní jejich opětovné otevření při selhání primární trasy (např. přerušením kabelu nebo výpadkem některého prvku po cestě).

Topologie je řízena prostřednictvím priorit. Každé zařízení může teoreticky být tzv. root, od kterého je topologie stavěna. Jako root je voleno zařízení s nejnižší prioritou, případně s nižší MAC adresou. V závislosti na cenách (prioritách) jednotlivých linek jsou některé z nich vybrány jako funkční, ostatní jsou v záložním stavu.

Určitou nevýhodou STP je poměrně dlouhá konvergence sítě v případě výpadku primární trasy nebo prvku, který je aktuálně zvolen jako root.

Lze doporučit následující věci:

- zvolit root tak aby bylo jasné, který prvek tuto funkci vykonává
- root by měl být zároveň nejstabilnější zařízení sítě (např. redundantně vybavený centrální přepínač)
- vypnout STP na všech portech na které jsou připojeny stanice (důvodem jsou problémy při přihlašování k Netware a problémy s DHCP).



## Síťová vrstva – adresování.

Adresace v počítačových sítích, nezávisle na typu protokolu, musí zajistit unikátnost adresy uzlu v rámci celé sítě. Tento problém je řešen **logickým rozdělením adres na část adresy sítě a adresy uzlu** (jde o jakousi analogii telefonních čísel, kde je koncový uzel jednoznačně určen dvojicí číslo předvolby a číslo koncové stanice). **Nejdůležitějším protokolem** se společně s rozvojem Internetu stal **protokol IP**. Nejde pouze o jeden protokol, ale o celou sadu protokolů. Tato sadu lze popsat na základě sedmivrstvého modelu OS. Z plejády protokolů se v této části budeme věnovat právě tomu jenž dal celé sadě jméno – **Internetwork Protokolu (IP)**. Jde o protokol 3. vrstvy OSI (síťové) a jeho **úkolem je zajistit adresaci a bezspořádanou přepravu datagramů v rámci síťového prostředí.**

### Adresování IPv4

V IP prostředí je konkrétní vyjádření **adresy ve formátu 4 jednobytových čísel oddělených tečkami**. Vypadá tedy následovně – x.x.x.x (např. 192.168.1.3). **Rozdělení na část adresy sítě a adresy uzlu** není úplně triviální jako u některých jiných protokolů (např. IPX, DECNet, Vines IP, ...). **Určující jsou nejvyšší bity prvního oktetu**. Podle nich se **adresy dělí do několika tříd** z nichž nejvýznamnější jsou **A, B a C** a případně **D**. V následující tabulce znamená **s část sítě a u část uzlů**.

třída IP adr.	nejvyšší bity prvního okte- tu adresy	formát	rozsah	počet sítí	počet uzlů
<b>A</b>	<b>0</b>	<b>s.u.u.u</b>	<b>1.x.x.x</b> až 126.x.x.x	126	16.777.214
<b>B</b>	<b>10</b>	<b>s.s.u.u</b>	<b>128.0.x.x</b> až 191.254.x.x	16.384	65.534
<b>C</b>	<b>110</b>	<b>s.s.s.u</b>	<b>192.0.0.x</b> až 223.254.254.x	2.097.150	254
D	1110	s.s.s.u	224.0.0.x až 239.254.254.254	pro multicast aplikace	
E	1111		240.0.0.x až 255.254.254.254	pro výzkum	

přičemž **nuly v adrese sítě sice nejsou striktně zakázány**, ale jejich **použití se nedoporučuje** – na síti se může vyskytnout zařízení, které je nepodporuje. **V adrese uzlu se nula stejně tak jako číslo 255 vyskytnout nesmí**. Adresní prostor 127 je rezervován pro **loopback** (testování komunikace, 127.0.0.1). Aby situace nebyla tak jednoduchá byl definován **pojem maska IP sítě**.

### Rezervované IP adresy.

Některé adresy jsou rezervované a nelze je použít pro adresování v síti. Adresy obsahují:

- 1. adresu sítě** (slouží pro identifikaci sítě, končí 0, např. 192.168.1.0)
- 2. adresu broadcastu** (zpráva se pošle všem stanicím v síti, končí 255, např. 192.168.1.255)

### Veřejné a privátní IP adresy.

V Internetu je třeba, aby každý měl jedinečnou IP adresu. Zabezpečuje to Network Information Center (InterNIC). Adresy se dělí na:

**Veřejné adresy:** jsou jedinečné, dva uzly (PC), které jsou připojené na veřejnou síť, nemohou mít stejnou IP adresu, protože veřejné adresy jsou globální a standardizované. Jsou přidělovány Internet service providerem (ISP). Rozsah IP adres se stával nedostatečný, proto byla vyvinutá nová adresní schémata jako classless interdomain routing (CIDR) a IPv6.

**Privátní IP adresy:** privátní sítě, které nejsou připojené do internetu, mohou používat jakékoli IP adresy, ale opět ve vnitřní síti musí být jedinečné. V tabulce jsou uvedené rozsahy privátních adres podle tříd.



Class	RFC 1918 internal address range
A	10.0.0.0 to 10.255.255.255
B	172.16.0.0 to 172.31.255.255
C	192.168.0.0 to 192.168.255.255

RFC vyčlenil 3 bloky adres pro privátní použití. Jsou vybrané z A, B a C tříd.  
Pro překlad privátních adres na veřejné slouží systém NAT.

## Network Address Translation

(**NAT**, česky *překlad síťových adres*, také **Network Masquerading** (*síťová maškaráda*), **Native Address Translation** (*nativní překlad adres*) nebo **IP Masquerading** (*IP Maškaráda*)) je způsob úpravy síťového provozu přes router přepisem výchozí a/nebo cílové IP adresy, často i změnu čísla TCP/UDP portu u průchozích IP paketů. K tomu patří i změna kontrolního součtu (u IP i TCP/UDP), aby změny byly brány v úvahu. NAT se většinou používá pro přístup více počítačů z lokální sítě na Internet pod jedinou veřejnou adresou (viz gateway). NAT ovšem může způsobit problémy v komunikaci mezi klienty a snížit rychlost přenosu.

Vznikl jako důsledek omezeného počtu veřejných IP adres (IPv4 má 32 bitů a část z nich je navíc rezervována pro speciální účely). Každý uživatel dnešního internetu nemůže mít adresu z vnějšího rozsahu, NAT umožňuje celou vnitřní síť ukrýt za adresu jedinou.

V typické konfiguraci používá lokální síť některý z rozsahů IP adres (192.168.x.x, 172.16.x.x - 172.31.x.x a 10.x.x.x). Router má přidělenou soukromou adresu, ale také je spojen s Internetem, buď jedinou veřejnou adresou (overloaded NAT), nebo několika veřejnými adresami, přidělenými poskytovatelem připojení k Internetu (Internet Service Provider). Jakmile jde paket z lokální sítě do Internetu, je jeho zdrojová adresa (soukromá) přeložena na veřejnou.

Router si uchovává základní data o každém aktivním spojení (adresu cíle, port). Když se vrátí odpověď na router, využije data získaná při odchozí fázi a určí kam na vnitřní síť je třeba odpověď zaslat. Pro systém na Internetu se jeví router jako zdroj i cíl komunikace.

Pakety posílané protokoly TCP a UDP obsahují kontrolní součet zahrnující i hlavičku s adresami, tudíž je třeba kromě přepsání IP adres také přepočítat kontrolní součet pro IP adresu novou.

## Definice

Překlad síťových adres je funkce, která umožňuje překládání adres. Což znamená, že adresy z lokální sítě přeloží na jedinečnou adresu, která slouží pro vstup do jiné sítě (např. Internetu), adresu překládanou si uloží do tabulky pod náhodným portem, při odpovědi si v tabulce vyhledá port a pošle pakety na IP adresu přiřazenou k danému portu.

NAT je vlastně jednoduchým proxy serverem. NAT může být softwarového typu (Nat32, Kerio Winroute firewall), nebo hardwarového typu (router s implementací nat).

## Vlastní komunikace

Klient odešle požadavek na komunikaci, směrovač se podívá do tabulky a zjistí, zdali se jedná o adresu lokální, nebo adresu venkovní. V případě venkovní adresy si do tabulky uloží číslo náhodného portu, pod kterým bude vysílat a k němu si přiřadí IP adresu

## Výhody

- umožňuje připojit více počítačů na jednu veřejnou IP adresu - řeší se tak nedostatek přidělených veřejných IP adres.
- zvyšuje bezpečnost počítačů připojených za NATem (potenciální útočník nezná opravdovou IP adresu)

## Omezení

Nemáte skutečné připojení k Internetu a tak vám budou fungovat jen ty aplikace, kterým to nevadí. Pokud patříte mezi uživatele, kteří si myslí, že WWW a Internet jsou synonyma, pak si nejspíš překladu vůbec nevšimnete, v opačném případě můžete narazit na situace, kdy vám některé aplikace nebudou fungovat. U některých to půjde vyřešit konfigurací snadno, u jiných obtížně a funkčnost může zůstat omezená, u některých to nepůjde vůbec.

NAT zvyšuje bezpečnost počítačů za ním připojených (útočník nezná strukturu sítě a nemůže se spojit přímo s konkrétním počítačem). NAT ovšem firewall nenahrazuje a existují způsoby, jak počítače za NATem napadnout.

## Druhy NAT

- **Network Address Port Translation** (NAPT, PAT, IP maškaráda), kdy dochází k mapování čísel portů. Několik strojů pak může sdílet jednu veřejnou IP adresu.

- **NAT jedna ku jedné** (basic NAT, static NAT), umožňující pouze překlad adres, nikoli mapování portů. Tato možnost vyžaduje IP adresu pro každé samostatné spojení.

## Omezení NATu

Počítače za NATem mají jistá omezení, nemohou běžně provozovat některé Internetové protokoly. Například služby vyžadující příchozí TCP spojení z vnější sítě. Případem problémových protokolů jsou například FTP nebo SIP, které přenášejí informaci s adresou navíc uvnitř paketu. Při překladu adres se musí pakety zpracovávat celé a to může být velice náročné, proto tyto protokoly nebývají podporovány.

Problémy se dají vyřešit několika způsoby:

- ALG (Application Layer Gateway), softwarový modul běžící na zařízení s NATem
- Techniky tunelování NATu užitím protokolů jako jsou STUN (Simple Traversal UDP through NAT), ICE (Interactive Connectivity Establishment) nebo TURN (Traversal Using Relay NAT)
- UPnP (Universal Plug and Play) nebo Bonjour (NAT-PMP) vyžadující kooperaci se zařízením s NAT

## DNAT

DNAT je technika pro transparentní změnu cílové IP adresy paketu na cestě a pro vykonání opačné funkce pro jakékoliv odpovědi. Tuto transformaci může vykonat kterýkoliv router umístěný mezi dvěma koncovými body. DNAT se obvykle používá při zveřejnění služby, umístěné v privátní síti, pro veřejně přístupnou IP adresu.

## SNAT

Použití termínu SNAT je u každé softwarové firmy jiné. To znamená, že každá firma má svou definici pro SNAT. Microsoft například používá tento název pro Bezpečný (Secure) NAT. Obecná definice je Zdrojový (Source) NAT, tedy protějšek Cílového NATu (DNAT).

## Informativně o adresování IPv6

IP verze 6 je protokol síťové vrstvy podle ISO/OSI, která má za úkol adresování a směrování v síti. IPv6 vznikl jako následník verze 4, důvodem k tomu byla snaha předejít vyčerpání adresního prostoru a začlenit do protokolu podporu nových služeb. IPv6 přináší větší adresový prostor, možnost autokonfigurace, více adres na jednom rozhraní, bezpečnostní mechanismy síťové komunikace a podporu mobility.

## Adresy a doručování

Adresy v IPv6 mají velikost 128b a zapisují se jako 8 čtveřic hexadecimálních čísel oddělených dvojtečkou. Je možné vynechat nuly zleva či čtveřici nul zkrátit na jednu. Navazující čtveřice nul je možné zkrátit na ::, ale vždy nejvýše jednou v jedné adrese. Příkladem zápisu IPv6 adresy je adresa fc00:0:0:0:0:0:90:0, kterou můžeme zkráceně zapsat jako fc00::90:0.

V URL se IPv6 adresa uvádí v hranatých závorkách, aby bylo možné rozlišit adresu a port, např.

[https://\[fc00:0:0:0:0:0:1234:1234\]:443/](https://[fc00:0:0:0:0:0:1234:1234]:443/).

Síťové prefixy se uvádí ve stejné notaci jako CIDR, tedy např. fe80::/10. IPv6 nezná síťové třídy jako IPv4.

## EUI-64 formát adres

IP adresu je možné přidělit na základě fyzické adresy rozhraní. V ethernetu se použije MAC adresa, do které se vkládá fffe za identifikátor společnosti. Sedmý nejvyšší bit pak udává, zda je adresa lokální či globální. Protože je možné MAC adresu změnit, definuje RFC 4941 způsob náhodného generování identifikátorů rozhraní platných jen po omezenou dobu.

## Doručování paketů

V IPv6 máme několik možností doručování paketů:

**Unicast** - Unicastová adresa určuje konkrétní rozhraní v síti.

**Multicast** - Adresa pro skupinu rozhraní. Jednotlivé unikátní adresy se přihlašují do multicastových skupin. Multicast s adresou all-hosts nahrazuje broadcast, který v IPv6 nenajdeme.

**Anycast** - Opět adresa pro skupinu rozhraní, paket se ale doručí jen jednomu ve skupině (typicky nejbližšímu podle směrování). Jedná se o novinku v IPv6.

### Adresování

Přehled vybraných adres a adresových rozsahů:

**::/128** - nespecifikovaná adresa

**::1/128** - loopback

**fe80::/10** - lokální linkové adresy, jedinečné v místní síti

**fc::/7** - unikátní lokální adresy, jedinečné typicky v rámci organizace

**ff00::/8** - multicastové adresy

**::ffff:0:0/96** - prefix IPv4 adres mapovaných do IPv6

**2002::/16** - adresa pro tunelování v IPv4

Lokální linkové adresy jsou přístupné pouze v místní síti (lince), využívají identifikátor rozhraní podle EUI-64 a jsou přístupné vždy. Unikátní lokální adresy mají podobný význam jako lokální linkové adresy - určují rozhraní v rámci sítě dané organizace, která může rozdělena na několik podsítí. Globální individuální adresy, celosvětově jedinečné tvoří zbytek adresního prostoru a jsou přidělovány podobně jako v IPv4 (poskytovatel dostane přidělen prefix, který dále rozděluje mezi své klienty).

### DNS a IPv6

IPv6 adresy se v DNS v dopředných zónách uvádí v zápnamech AAAA. Reverzní vyhledávání probíhá pod ip6.arpa, v záznamu je obráceno pořadí šestnáctkových číslic celé adresy a každé z nich reprezentuje subdoménu.

### Přechod mezi IPv4 a IPv6

**Dual stack** - Zařízení v síti podporuje oba protokoly, pro každou verzi má oddělený prostor.

**Tunelování** - IPv4 paket je vložen do paketu IPv6 a odeslán IPv6 sítí.

**Autotunelování** - IPv6 adresa 2002:xxxx:xxxx::/48 se v IPv4 síti směruje jako IPv4 adresa uvedená v xxxx:xxxx. O převod mezi sítěmi se starají brány (6 to 4 relay).

**Proxy a překlad adres** - Pro přístup ke službám IPv6 ze sítě IPv4 je možné využít i proxy překlad adres. Implementace obou řešení je však náročná a nemusí podporovat všechny služby sítě.

### Formát datagramu v IPv6

Formát datagramu má v IPv6 obvyklou podobu: nejprve hlavička, za ní pak vlastní nesená data. Ovšem hlavička stojí rozhodně za pozornost. Přestože se délka adres prodloužila na čtyřnásobek, celkově má hlavička ve srovnání s IPv4 jen dvojnásobnou délku: 40 bajtů, z toho 32 B tvoří adresy odesílatele a příjemce.

Zajímavý je přístup k různým rozšiřujícím informacím. V IPv4 byly přímo součástí hlavičky, jako její nepovinné položky. V důsledku toho měla hlavička proměnlivou délku. IPv6 řeší problém zcela odlišně. Základní hlavička byla maximálně zjednodušena a obsahuje jen nejnutnější položky. Její složení i délka je konstantní.

Položka nazvaná *další hlavička* (*next header*) identifikuje, co následuje za základní hlavičkou. Může se jednat o TCP nebo UDP data, pokud datagram vystačí se základní hlavičkou. Je-li třeba přidat doplňující informaci, vloží se v podobě rozšiřující hlavičky. Každá z rozšiřujících hlaviček má svou položku *další hlavička*. Z jejich hodnot pak vznikne řetěz, který postupně sděluje "za základní hlavičkou je tahle rozšiřující, pak támhle, pak ještě takováhle a pak následují data pro TCP.

Pořadí rozšiřujících hlaviček je předepsáno tak, aby nejprve byly umístěny ty, které potenciálně mohou zajímat i směrovače, jimiž datagram prochází. Celé toto uspořádání má jediný cíl: aby směrovače měly co nejlehčí práci. Prozkoumají základní hlavičku a hodnotu v položce *další hlavička*. Pokud je dotyčná hlavička určena

pro ně, zařadí ji do zpracování a zajímají se o další. Jakmile narazí na první rozšiřující hlavičku, která není pro ně, mohou zkoumání paketu ukončit - nic dalšího zajímavého je již nečeká.

Podívejme se, jak vypadá základní hlavička IPv6 datagramu:



**První čtveřice bajtů** slouží vedle identifikace verze protokolu k realizaci pokročilejších směrovacích technik.

**Třída provozu** umožňuje specifikovat požadavky na vlastnosti sítě, které daný datagram má. Prostřednictvím **značky toku** se pak identifikuje tok - proud datagramů od jednoho odesílatele ke stejnému cíli se stejnými vlastnostmi. Příslušnost ke stejnému toku by měla usnadnit směrování, zejména ve spolupráci se spojovanými technologiemi, jako je třeba ATM.

Ani jeden z těchto pokročilejších prvků zatím nebyl detailně definován. Je to zřejmě osud, protože v hlavičce IPv4 také je položka *typ služby (TOS)* podobného významu, u které bylo přislíbeno pozdější upřesnění. Nikdy k němu pořádně nedošlo.

Na druhém řádku najdete celkem očekávatelné údaje: **délku přenášených dat**, avizovanou identifikaci typu **další hlavičky** a konečně maximální počet směrovačů, kterými datagram ještě smí projít (**max. skoků**, analogie *TTL* z IPv4). Každý směrovač ji sníží o jedničku a dojde-li do nuly, je povinen datagram zahodit a odeslat odesílateli ICMPv6 zprávu o tomto skutku. Je to klasický způsob ochrany před zacyklením při směrování.

A dále už následují jen adresy: **zdrojová a cílová**. Všimněte si, že **hlavička neobsahuje kontrolní součet**. Zdržoval by a v praxi není potřebný (o kontrolu dat se postarají nižší vrstvy).

Ve srovnání s IPv4 zmizely také položky věnované fragmentaci. IPv6 se snaží, aby k ní pokud možno nedocházelo. Proto by komunikující partneři měli měřit MTU (maximální velikost paketu) celé přenosové trasy a posílat dostatečně malé pakety. Pokud k ní přece jen dojde, přibalí se rozšiřující hlavička *Fragmentace*. Fragmentovat smí jen odesílatel dat.

## Přehled rozšiřujících hlaviček

Volby pro všechny	0	informace zajímavé pro každého po cestě (např. upozornění směrovače, že paket nese data, která by jej mohla zajímat)
Směrování	43	datagram musí projít předepsanou cestou
Fragmentace	44	při fragmentaci paketu nese informace nutné pro jeho složení do původní podoby
Šifrování obsahu (ESP)	50	obsah datagramu je zašifrován, ESP hlavička nese odkaz na parametry pro dešifrování
Autentizace (AH)	51	data pro ověření totožnosti odesilatele a původnosti obsahu
Poslední hlavička	59	nic dalšího nenásleduje
Volby pro cíl	60	informace určené příjemci datagramu (např. domácí adresa mobilního uzlu)
Mobilita	135	pro potřeby komunikace s mobilními zařízeními

## Adresace IPv4 v podsítích, masky sítě.

**Masky** zajišťují mechanismus jak jednu síť rozdělit do **logických podsítí**.

*Příklad: Vezměme třídu A – je připravena pro 126 sítí, každá s 16.777.214 uzly. Bez maskování má k dispozici jednu ohromnou broadcastovou doménu.*

S použitím masek si může síť rozdělit do mnoha podsítí. Část sítě přidaná maskou se nazývá podsít (subnet).

Princip masek vychází z předpokladu, že tam, kde je v binárním vyjádření masky jednička, tam je síť. Tam kde je nula, je uzel.

**Podívejme se nejprve na přirozené masky jednotlivých rozsahů:**

přirozená maska                      binární vyjádření masky

**třída A**            255.0.0.0            11111111.00000000. 00000000. 00000000

(má rozsah sítí dán prvním oktetem)

**třída B**            255.255.0.0            11111111. 11111111. 00000000. 00000000

(má rozsah dán prvními dvěma oktety)

**třída C**            255.255.255.0            11111111. 11111111. 11111111. 00000000

(má rozsah dán prvními třemi oktety)

Každý bit masky má hodnotu, která se podle počtu bitů načítá (ještě jednou upozorňuji na kontinuitu zleva!):

bit	7	6	5	4	3	2	1	0
hodnota	128	64	32	16	8	4	2	1
maska	128	192	224	240	248	252	254	255

Pomocí masek jsme schopni oblast sítě roztáhnout na úkor oblasti uzlů.

Viz. příklad:

dekadické vyjádření

binární vyjádření

adresa    10.1.1.1            00001010.00000001.00000001.00000001

maska     255.255.0.0            11111111.11111111.00000000.00000000

Jedná se o nejjednodušší typ maskování, tedy maskování vyšší třídy přirozenou maskou nižší třídy. V tomto případě Áčko Běčkem. Adresa sítě je v tomto případě 10.1, adresa uzlu je 1.1.

Vedle přirozených masek existují masky nepřirozené. To jsou ty, které nemají v oktetu samé jedničky, ale jsou zprava doplněny nulami. Jedničky musí být v masce zleva bez přerušení!

Slash format	/25	/26	/27	/28	/29	/30	N/A	N/A
Mask	128	192	224	240	248	252	254	255
Bits borrowed	1	2	3	4	5	6	7	8
Value	128	64	32	16	8	4	2	1
Total Subnets		4	8	16	32	64		
Usable Subnets		2	6	14	30	62		
Total Hosts		64	32	16	8	4		
Usable Hosts		62	30	14	6	2		

### Základní pravidla pro použití masek jsou:

- jedničky musí být v masce zleva bez přerušení;
- subnet – čísla podsítí složená ze samých nul nejsou doporučena (ale lze je použít – pozor u některých prostředků se musí povolit!);
- subnet – čísla podsítí složená ze samých jedniček nejsou povolena;
- síťové adresy složené ze samých jedniček nebo samých nul nejsou povoleny;
- adresy uzlů složené ze samých jedniček nebo samých nul nejsou povoleny.

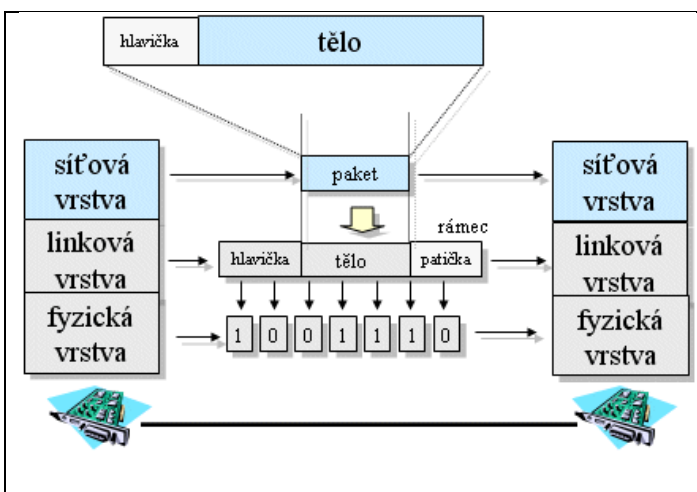
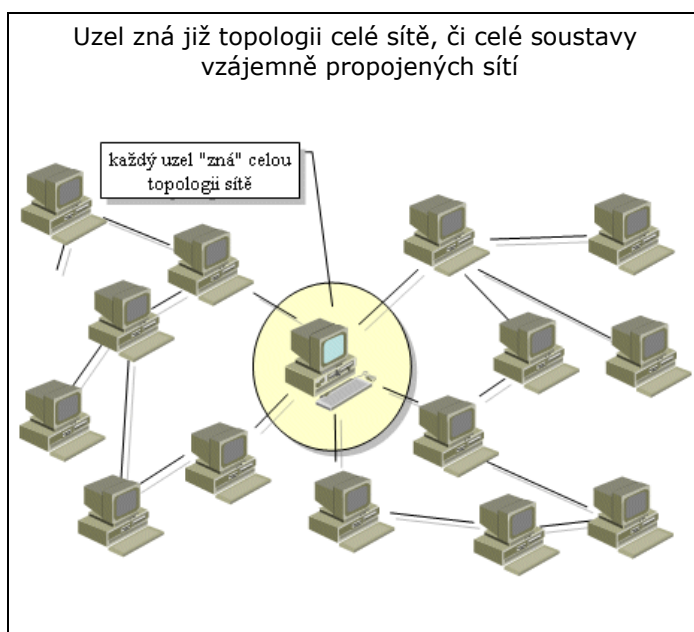
### Funkce síťové vrstvy.

Zná topologii celé sítě, či celé soustavy vzájemně propojených sítí, a díky tomu dokáže hledat a nacházet cesty ke kterémukoli cílovému uzlu.

Samotnému hledání cesty ke koncovému příjemci, které síťová vrstva zajišťuje, se říká **směrování** (anglicky **routing**).

Jde tedy spíše o logickou ("rozhodovací") funkci, jejímž výsledkem je rozhodnutí o konkrétní cestě pro přenos. Jak záhy uvidíme, postup tohoto rozhodování může být založen na opravdu různých principech., přístupech i metodách. Podle toho se pak také hovoří například o centralizovaném směrování, izolovaném či distribuovaném směrování, či směrování hierarchickém atd. Samotné rozhodnutí o volbě cesty, a s ním i o směru dalšího přenosu (z pohledu každého uzlu "po cestě"), je ale třeba také konkrétně naplnit. Tedy vzít příslušný blok dat (paket, jak se mu říká na úrovni síťové vrstvy), a přenést jej k dalšímu "přestupnímu" uzlu, případně již k samotnému cílovému adresátovi (pokud jde již o konec celé přenosové cesty). Tomu se obvykle říká "forwarding" (v doslovném překladu jakési "předání dál").

Síťová vrstva ovšem sama o sobě nic nepřenáší. To je pouze iluze, vytvořená vrstevnatým charakterem dnešních sítí. Ve skutečnosti totiž síťová vrstva předá svůj paket k přenosu bezprostředně nižší vrstvě, tj. vrstvě linkové. Ta vloží síťový paket do





svého rámce (linkového rámce), a ten pak přenáší. Ovšem ani ona jej ve skutečnosti sama nepřenáší, nýbrž nechává si jeho jednotlivé bity přenést od vrstvy fyzické.

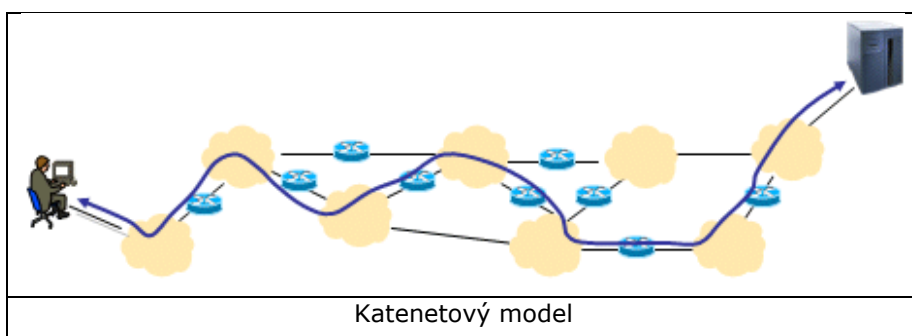
## Typy směrování:

### Přímé směrování

Jak jsme si již řekli, linková vrstva, které vrstva síťová předává své pakety k přenesení, "dohlédne" jen ke svým bezprostředním sousedům, se kterými má přímé spojení (viz výše). V praxi jsou těmito sousedy všechny uzly, které se nachází ve stejné síti. Pokud se tedy přenáší nějaký síťový paket mezi dvěma uzly v rámci stejné sítě, pak má samotná síťová vrstva značně usnadněnou práci, protože nemusí hledat žádné mezilehlé (resp. přestupní) uzly, přes které by měl paket přejít. Takový paket proto síťová vrstva pouze předá linkové vrstvě, která již je schopna jej doručit cílovému adresátovi (neboť ten je v jejím přímém dosahu). Z pohledu síťové vrstvy se tomu říká "přímé směrování", a zde skutečně má síťová vrstva minimum práce a žádné "přemýšlení".

### Nepřímé směrování

Při tzv. nepřímém směrování, provádí síťová vrstva rozhodování, které jsme si výše označili jako samotné směrování (routing). To, co síťová vrstva při nepřímém směrování hledá, je uzel, vedoucí z dané sítě "ven", do jiné sítě. Tedy jakýsi "přestupní" uzel, resp.

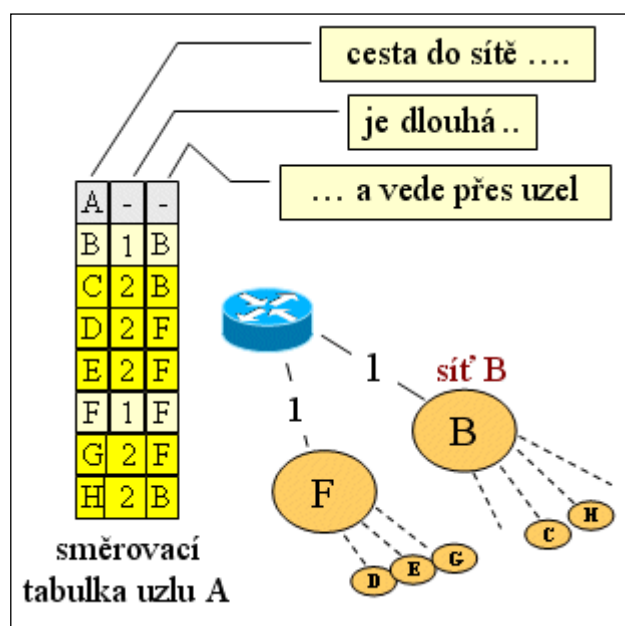


"propojovací" uzel (uzel, který vzájemně propojuje dvě různé sítě, či dokonce více takovýchto sítí). Takovému uzlu se říká směrovač (router), a v každé síti, která není zcela izolována od svého okolí, je nejméně jeden. Odpovídá to tzv. katenetovému modelu, v rámci kterého jsou jednotlivé sítě propojeny, pomocí směrovačů, do celé soustavy vzájemně propojených sítí. Na takovémto modelu je vybudován například dnešní Internet, ale obecně z něj vychází prakticky všechny počítačové sítě.

## Směrovací tabulky

K tomu, aby se směrovače mohly snáze a rychleji rozhodovat, udržují si potřebné (směrovací) informace v tzv. směrovacích tabulkách. Jejich obsah, resp. organizace, se může lišit, ale v nejjednodušším případě je takový, jak jej ukazuje následující obrázek: každá jednotlivá položka směrovací tabulky je vyhrazena jedné cílové síti. Součástí položky je pak i informace o tom, jak je cílová síť daleko, a hlavně kudy (kterým "odchodem" směrem) je třeba paket předat, aby se dostal blíže k cílové síti (neboli: kudy pokračuje cesta k této cílové síti).

S popisovaným obsahem směrovacích tabulek souvisí ještě jedna důležitá skutečnost: směrování, jako rozhodování o dalším směru přenosu, vychází z příslušnosti



cílového uzlu k určité síti. Jinými slovy: směruje se podle sítě, a nikoli podle konkrétního uzlu. Proto také jsou jednotlivé položky směrovacích tabulek (alespoň v TCP/IP) vyhrazeny celým cílovým sítím, a nikoli jednotlivým uzlům v těchto cílových sítích. Jinak by totiž počty položek ohromně narostly. Odpovídá to ostatně i roz-



dílu mezi přímým a nepřímým směrováním: po celé trase přenosu probíhá nepřímé směrování, a v každém kroku se zajišťuje přenos do další sítě jako takové (jako celku). Teprve v poslední (cílové) síti se již nejedná o nepřímé směrování, ale o směrování přímé (v rámci dané sítě). Zde už tedy směrovací tabulka není zapotřebí, protože doručení konkrétnímu uzlu v cílové síti zajistí zdejší linková vrstva.

Existence směrovacích tabulek se přitom netýká pouze směrovačů. Mají je i koncové uzly, protože i ony se podle nich musí rozhodovat při odesílání, přes který "odchází" směrovač data poslat. Jejich směrovací tabulky ale bývají v praxi podstatně menší. Základem jejich obsahu je informace o existenci výchozí brány, kterou jsme si popisovali výše. Další položky pak do směrovací tabulky koncového uzlu přibývají obvykle postupně, na základě "upozornění" od směrovače, že správný směr vede jinudy (přes jiný směrovač, viz výše). Zobrazení tabulky v CMD okně příkazem „route print“

### Adaptivní a neadaptivní směrování

Kdo a jak stanovuje a aktualizuje obsah směrovacích tabulek směrovačů? Zde už to totiž je komplikovanější, než u koncových uzlů. V úvahu zde připadají dvě základní možnosti:

"**neadaptivní**": obsah směrovacích tabulek (směrovačů) je dán a nemění se. Pak jej může jednorázově stanovit například správce sítě, a veškeré směrování bude následně fungovat vždy stejně.

"**adaptivní**": směrovací tabulky směrovačů jsou na počátku nějak nastaveny, ale jejich obsah se průběžně aktualizuje, tak aby odrážel topologii soustavy sítí a reagoval na její změny.

Každá z těchto dvou variant samozřejmě má své výhody, ale i své nevýhody.

**Neadaptivní varianta** nemá žádnou režii na své průběžné udržování (když k němu nedochází), a je také více odolná vůči eventuálním útokům, které by se snažily mechanismus směrování nějak nabourat. Proto se používá například tam, kde jsou zvýšené požadavky na bezpečnost. **Nevýhodou neadaptivní varianty je ale její neschopnost reagovat na změny v síti a její topologii.** To při zvýšených požadavcích na bezpečnost může být i výhodou, ale obecně je to nevýhodou, kvůli které lze tuto variantu doporučit jen tam, kde lze předpokládat zcela ojedinělé změny v topologii.

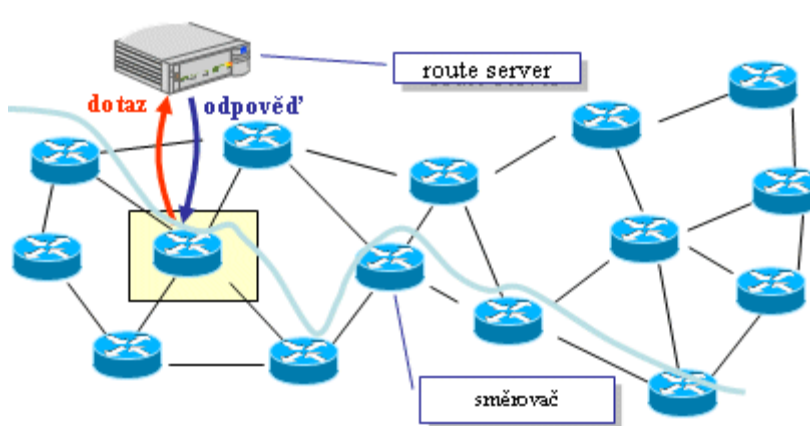
**Adaptivní varianta** přímo počítá s tím, že bude na eventuální změny v topologii reagovat. Takže ji lze využít i tam, kde ke změnám dochází relativně častěji. Důležité ovšem je, že s touto adaptivností může být spojena i poměrně velká režie, na výměnu aktualizací informací (informací o změnách v síti). Zde velmi záleží na tom, jaké varianty směrování (a výměny směrovacích informací) jsou konkrétně použity. Pojdme se proto podívat, jaké možnosti zde připadají v úvahu.

### Centralizované směrování

Začneme nejprve poněkud specifickou variantou, označovanou jako centralizované směrování. Jde o řešení, v rámci kterého veškeré rozhodování (o volbě nejvhodnější cesty) provádí jeden centrální prvek, zatímco všechny směrovače již jen mechanicky naplňují jeho rozhodnutí. Lze si to představit také tak, že jednotlivé směrovače přestanou samy přemýšlet, a kdykoli neví co a jak mají dělat, zeptají se centrálního prvku. Tomu

se říká "**route server**", což by se dalo rozepsat jako "**server, poskytující jako svou službu informace o cestách**". Na konkrétní dotaz tento server odpoví, a směrovač se podle toho zachová. A aby se při dalším paketu nemusel ptát hned znovu, a příliš tak nezatěžoval route server, každou odpověď si po nějakou dobu pamatuje ve své vyrovnávací paměti a podle ní se řídí. Tato doba přitom musí být volena velmi pečlivě. Kdyby

Představa centralizovaného směrování



totiž byla příliš krátká, pak by se směrovač ptal příliš často a příliš by zatěžoval route server. Naopak, když bude moc dlouhá, hrozí nebezpečí že směrovač nebude včas reagovat na nějakou změnu v topologii. Stále totiž bude vycházet z předchozí odpovědi route serveru, místo toho aby si vyžádal odpověď novou, která by již odrážela příslušnou změnu.

Výhodou centralizovaného směrování je nulová režie na aktualizací informace, kterými by se jednotlivé směrovače jinak vzájemně informovaly o změnách. Další významnou výhodou je i soustředění veškeré "inteligence" do jednoho místa, což usnadňuje systémovou správu. Nevýhodou je, že zde vzniká "klíčové místo", s jehož výpadkem se celá soustava vzájemně propojených sítí stává nefunkční - jednotlivé směrovače se pak vůbec nedozví, do a jak mají dělat.

### Izolované směrování

Další variantou směrování, vedle centralizovaného, je směrování označované jako izolované. To proto, že zde už se rozhodují jednotlivé směrovače, ale skutečně jen samy za sebe, aniž by přitom spolupracovaly s ostatními. Proto přívlastek "izolované". Jde ovšem o celou skupinu různých dílčích variant, mezi které patří mj.:

- a) záplavové směrování
- b) směrování metodou horké brambory
- c) náhodné směrování
- d) metoda zpětného učení
- e) a další.

Například záplavové směrování, které je v praxi relativně často používané, funguje skutečně tak jak naznačuje jeho název: každý směrovač rozešle přijatý paket do všech směrů, kromě toho odkud přišel. Tím vzniká jakási záplava, která dříve či později dorazí na místo svého správného určení. Současně je ale nutné správně a korektně eliminovat duplicitní pakety, které při záplavovém rozesílání vznikají, aby na místo svého určení dorazil každý paket jen jednou. Výhodou je velká robustnost - pokud existuje alespoň nějaká cesta k cíli, je tímto způsobem nalezena. A vlastně bez toho, aby směrovače vůbec potřebovaly nějaké směrovací tabulky.

Pro zajímavost se zmiňme ještě o metodě horké brambory. Ta se začíná používat (jako náhradní varianta) v situaci, kdy jinak používané směrování se dostává do problémů a směrovači hrozí zahlcení. Když se v některém odchozím směru začínají hromadit pakety, které směrovač nestíhá odeslat, může nastoupit právě směrování metodou horké brambory - směrovač volí odchozí směr nikoli podle toho, kudy by měl paket správně pokračovat, ale podle toho, který odchozí směr je právě nejméně vytížený (tj. kde se "horké brambory" dokáže co nejrychleji zbavit). Když se pak nebezpečí zahlcení podaří odvrátit, směrovač se zase vrací ke svému původnímu způsobu směrování.

### Distribuované směrování

V praxi nejčastěji používané varianty směrování však spadají do kategorie "distribuovaných". To proto, že zde není žádný centrální prvek, který by o volbě směru rozhodoval (jako je tomu u centralizovaného směrování), ale celé rozhodování je rozděleno (distribuováno) mezi jednotlivé uzly, které na něm spolupracují (což je zase odlišnost od izolovaného směrování).

Jednou z variant distribuovaného směrování je takové, které je v angličtině označováno jako "vector-distance". To proto, že sousední směrovače si mezi sebou vyměňují celé své směrovací tabulky, i s jejich obsahem (viz výše). Každou položku směrovací tabulky si přitom lze představit jako vektor, který říká že z určitého směrovače se lze dostat do konkrétní cílové sítě takovým a takovým směrem, a že tato síť je tak a tak daleko. Když se pak nějaký směrovač dozví, ze směrovací tabulky svého souseda, že ten je vzdálen od určité cílové sítě X na Y jednotek, připočítá si k tomu svou vzdálenost od souseda, a vyjde mu že on se do sítě X dostane (přes svého souseda) se vzdáleností Y+1 jednotek (například). Právě toto je princip fungování varianty "vector distance", která je v praxi (v sítích TCP/IP) realizována například protokolem RIP (Routing Information Protocol). Ten je ale již poměrně staršího data, a v praxi se tolik nepoužívá.

Používanějším je **protokol OSPF** (Open Shortest Path First), který ale spadá pod jinou variantu a to "**link state**". To proto, že zde už si jednotlivé **směrovače** neposílají mezi sebou žádné vektory se vzdálenostmi, resp. celé své směrovací tabulky. Místo toho si **posílají pouze informace o tom, že někde existuje nějaké spojení mezi dvěma uzly (směrovači), a je funkční**. Každý směrovač - uzel rozešle do celé soustavy vzájemně propojených sítí (obvykle pomocí záplavového směrování) informaci o tom, kdo jsou jeho sousedé a zda jsou pro něj dosažitelní (zda spojení mezi nimi funguje). Ostatní směrovače, když takto "posbírají" informace o sousedech všech ostatních směrovačů, získají ucelenou představu o skutečné topologii celé soustavy sítí, a mohou si také samy vypočítat nejvhodnější cestu, odkudkoli kamkoli. Důležité také je, že aktualizací informací pak stačí posílat jen při nějaké změně (výpadku spojení, či naopak zřízení nějakého nového spojení mezi dvěma směrovači). Díky tomu je režie na průběžné šíření aktualizací informací výrazně menší, než u variant "vector distance" - a tak je směrování "link state" (s protokolem OSPF) v praxi použitelné i pro podstatně větší sítě, než první varianta (s protokolem RIP).

**Border Gateway Protocol (BGP)** je dynamický směrovací protokol používaný pro směrování mezi autonomními systémy (AS). Je základem propojení sítí různých ISP v peeringových uzlech. Je také nazývaný **exterior protocol**. **Směrování mezi autonomními systémy má charakteristické požadavky, které se nevyskytují v interním směrování. Směrovací tabulky obsahují stovky tisíc záznamů, nejdůležitějším kritériem nebývá vzdálenost, ale posuzují se nastavitelné parametry zohledňující například cenu a dodatečná pravidla aplikovaná v závislosti na zdroji, cíli, seznamu tranzitních autonomních systémů a dalších atributech.**

Vzhledem k velkému počtu záznamů se v případě změn v topologii vyměňují pouze informace o změnách, nikoliv celé směrovací tabulky jako je tomu v případě protokolu RIP.

**Protokol IGRP** patří do rodiny protokolů „distance vector“, a **vnitřních protokolů** (Interior Gateway Protocol (IGP)). **IGRP používá kompozitní metriku vypočítanou z několika položek, jako jsou zpoždění sítě (delay), šířka pásma (bandwidth), spolehlivost (reliability) a zatížení (load).** Správce sítě může nastavovat závažnost faktorů pro každou z těchto metrik, ačkoli zásah do těchto nastavení se musí dělat s velkou opatrností. IGRP poskytuje širokou škálu nastavení metrik. Spolehlivost a vytížení může například dosahovat jakékoliv hodnoty mezi 1 a 255; šířka pásma dosahuje hodnot odrážejících rychlosti od 1200 bps až do 10 Gbps, přičemž zpoždění dosahuje hodnot mezi 1 a 224. Tohle velké rozmezí metriky jsou dále doplněny sérií uživatelsky definovanými konstantami, které umožňují správci sítě ovlivňovat výběr směru cesty. Tyto nastavení jsou hashovány pro případ metriky, a jiných algoritmů, které poskytuje jednoduchá kompozitní metrika. Správce sítě může ovlivnit výběr cesty nastavením vyšší či nižší váhy specifické metriky. Toto přizpůsobení umožňuje správci doladit IGRP automatickou volbu cesty.

**Protokol EIGRP** byl vytvořen firmou CISCO jako rozšíření protokolu IGRP

# TRANSPORTNÍ VRSTVA

Transportní vrstava je první vrstvou, se kterou se setkáme pouze v koncových uzlech sítě, ale nikoli v jejích vnitřních uzlech (ve směrovačích). Však také hlavním úkolem transportní vrstvy je zajišťovat vzájemnou komunikaci koncových uzlů (tzv. end-to-end komunikaci). Je také první vrstvou, která v rámci uzlu rozlišuje jednotlivé entity (procesy), prostřednictvím tzv. portů. Nižší vrstvy se na jednotlivé uzly dívají vždy jako na dále nedělitelný celek.

## Architektura TCP/IP (na obrázku)

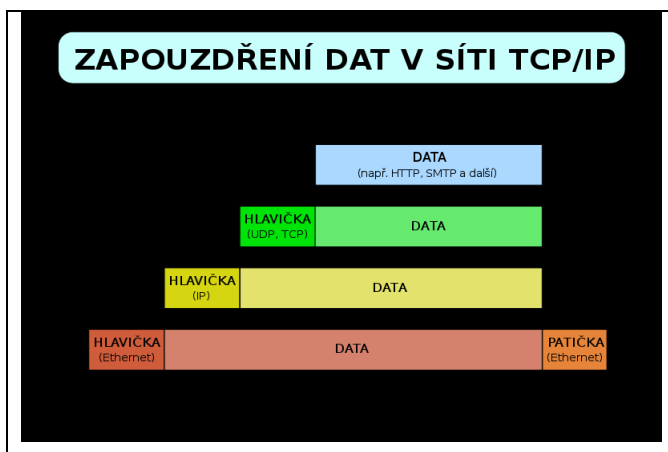
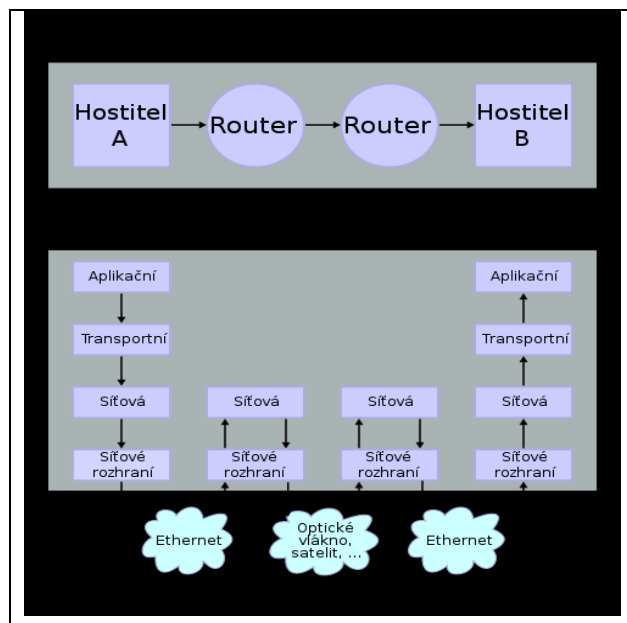
Vrstvy TCP/IP zajišťující přenos mezi dvěma hostiteli prostřednictvím dvou routerů. Vzhledem ke složitosti problémů je síťová komunikace rozdělena do tzv. vrstev, které znázorňují hierarchii činností. Výměna informací mezi vrstvami je přesně definována. Každá vrstva využívá služeb vrstvy nižší a poskytuje své služby vrstvě vyšší. Celý význam slova TCP/IP je **Transmission Control**

**Protocol/Internet Protocol** (primární transportní protokol - TCP/protokol síťové vrstvy - IP).

Komunikace mezi stejnými vrstvami dvou různých systémů je řízena **komunikačním protokolem** za použití spojení vytvořeného sousední nižší vrstvou.

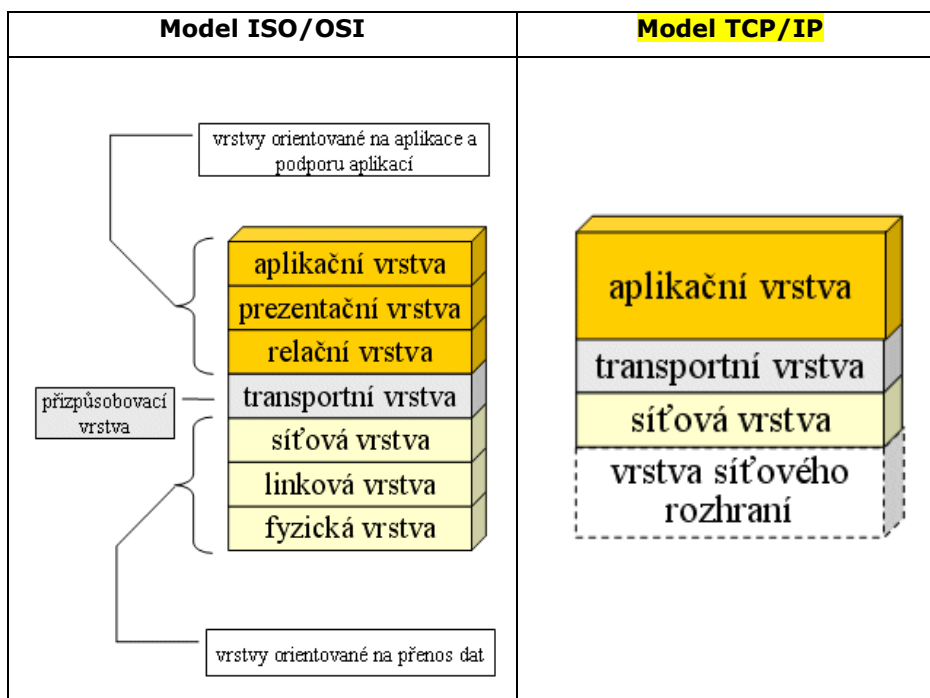
Architektura umožňuje výměnu protokolů jedné vrstvy bez dopadu na ostatní.

Příkladem může být možnost komunikace po různých fyzických médiích - ethernet, optické vlákno, sériová linka.



## Model TCP/IP

### Porovnání síťových modelů:



**Architektura modelu TCP/IP** je členěna do čtyř vrstev (na rozdíl od referenčního modelu OSI se sedmi vrstvami):

- aplikační vrstva (*application layer*)
- transportní vrstva (*transport layer*)
- síťová vrstva (*network layer*)
- vrstva síťového rozhraní (*network interface*)

### **Vrstva síťového rozhraní**

Nejnižší vrstva umožňuje přístup k fyzickému přenosovému médium. Je specifická pro každou síť v závislosti na její implementaci. Příklady sítí: Ethernet, Token ring, FDDI, X.25, SMDS.

### **Síťová vrstva**

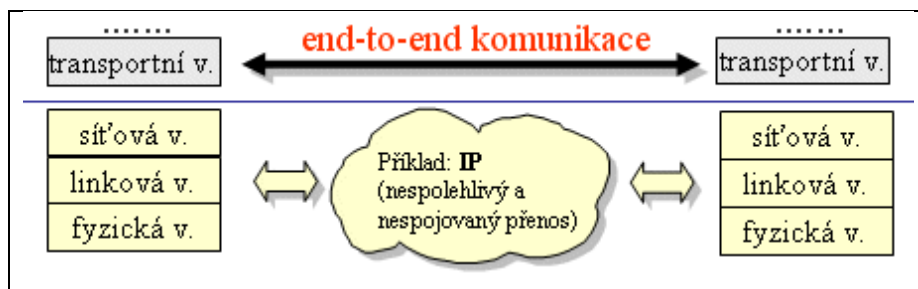
Vrstva zajišťuje především síťovou adresaci, směrování a předávání datagramů. Protokoly: IP, ARP, RARP, ICMP, IGMP, IGRP, IPSEC. Je implementována ve všech prvcích sítě - směrovačích i koncových zařízeních.

### **Transportní vrstva v modelu TCP/IP**

V rámci síťového **modelu TCP/IP** transportní vrstva samozřejmě existuje také, ale je teprve druhou (počítáno odshora), resp. třetí vrstvou (odspodu). Jde o důsledek skutečnosti, kterou bychom měli dobře znát již z předchozích dílů tohoto seriálu: že zatímco referenční model ISO/OSI má sedm vrstev, TCP/IP má pouze čtyři vrstvy. A také s nimi vystačí. Navíc svou nejnižší vrstvu, vrstvu síťového rozhraní, TCP/IP samo nijak nedefinuje (ale používá zde řešení pocházející odjinud). Ani menší počet vrstev v TCP/IP však nemění nic na tom, že transportní vrstva je první vrstvou (počítáno odspodu), která je přítomná až v koncových uzlech, a nikoli ve vnitřních uzlech sítě (ve směrovačích). Vzhledem k tomu má na starosti vzájemnou komunikaci koncových uzlů, pro kterou se i češtině vžil označení, pocházející z angličtiny: end-to-end komunikace (doslova: komunikace "konec-konec").

### **Představa end-to-end komunikace**

To, aby se transportní vrstva mohla soustředit na vzájemnou komunikaci koncových uzlů, jí umožňuje bezprostředně nižší vrstva - vrstva síťová.



### **Transportní vrstva z pohledu modelu ISO/OSI**

Referenční model ISO/OSI a síťový model TCP/IP se liší i v tom, jakou mají představu o fungování své síťové vrstvy. **RM ISO/OSI** má blíže k "telekomunikačnímu paradigmatu", v rámci kterého vítězí představa inteligentní a na funkce bohaté sítě ("chytré sítě"), a třeba i jednoduchých ("hloupých") koncových uzlů. Součástí této představy je i to, že přenosy dat na všech úrovních (vrstvách) by měly probíhat spojovaným způsobem, a tedy s navazováním spojení mezi odesílatelem a příjemcem. Dále byla představa, že přenosy dat na jednotlivých vrstvách by měly být řešeny jako spolehlivé, a tedy se starat o nápravu toho, co se pokazí - jak eventuelní ztráty dat, tak i eventuelní výpadky spojení. Ovšem ne vždy bude tato náprava dokonalá (stoprocentní). Proto se počítá s tím, že tato "náprava" bude pokračovat i na vyšších vrstvách, které se také budou snažit o zajištění (lepší, resp. vyšší) spolehlivosti. Autoři ISO/OSI rozhodli zavést **tři různé varianty (kategorie) síťových protokolů**:

**kategorie A:** pro prostředí, kde dochází k minimálním (žádným) ztrátám paketů a minimálním (žádným) výpadkům spojení

**kategorie B:** pro minimální (žádné) ztráty paketů, a občasné výpadky spojení

**kategorie C:** pro prostředí kde jsou občasné ztráty paketů a občasné výpadky spojení

Od transportní vrstvy se pak ve světě ISO/OSI očekává, že bude zachovávat spojovaný způsob fungování, a také že se bude snažit ještě lépe kompenzovat eventuelní výpadky spojení a ztráty dat. Dokonce se od ní očekává i určitá optimalizace, v tom smyslu že se může snažit navazovat více transportních spojení skrze jedno jediné síťové spojení. Motivace pro tuto optimalizaci pochází z prostředí veřejných datových sítí, kde se platilo za zřizování každého jednotlivého (síťového) spojení. Takže schopnost využít jedno takové spojení pro více spojení transportních mohla ušetřit náklady.

Celkově pak transportní vrstva ISO/OSI předpokládala pět různých variant (tříd) transportních protokolů, označovaných TP0 až TP4:

**třída TP0:** je pouze jednoduchou nadstavbou nad síťovým protokolem kategorie A, nemění jeho vlastnosti

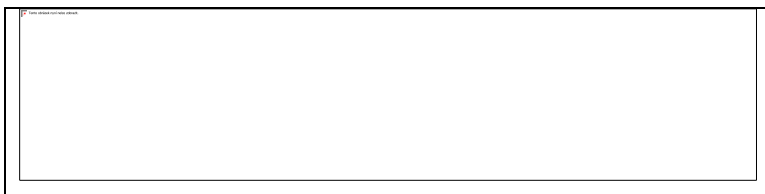
**třída TP1:** je nadstavbou nad síťovým protokolem kategorie B, omezuje případné výpadky spojení

**třída TP2:** je nadstavbou nad A, dokáže využít jedno síťové spojení pro více transportních spojení

**třída TP3:** je nadstavbou nad B, omezuje případné výpadky spojení a dokáže využít jedno síťové spojení pro více transportních spojení

**třída TP4:** je nadstavbou nad C, omezuje případné výpadky spojení a ztráty paketů

Celou představu ilustruje obrázek.



## Transportní vrstva z pohledu modelu TCP/IP.

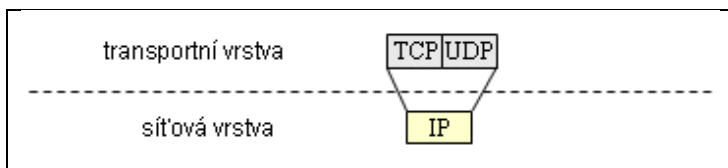
Ve světě TCP/IP se uplatňuje "počítačové" paradigma, volající po "hloupé síti a chytrých uzlech". Tedy po maximálním zefektivnění přenosové části sítě, která bude nabízet jen minimum funkcí v co nejjednodušším provedení (proto "hloupá síť"), ale zato je bude realizovat velmi efektivně (rychle). O všechno ostatní, pokud je to vůbec požadováno, by se měly starat až koncové uzly, na úrovni transportní (nebo aplikační) vrstvy, a nikoli přenosová část sítě (na úrovni vrstvy síťové).

Konkrétním příkladem toho, co je ve světě TCP/IP považováno za něco "ne zcela nezbytného", čím by se síťová vrstva neměla zatěžovat a zpomalovat, je zajištění spolehlivosti přenosů. Tedy kompenzace (náprava) případných ztrát paketů, pokud k nim vůbec dojde. Je-li vůbec spolehlivost přenosů požadována, měly by si ji zajistit koncové uzly samy. Vzhledem k celému paradigmatu ("hloupá síť, chytré uzly") k tomu mají lepší předpoklady než síť jako taková, resp. její síťová vrstva.

Společným důsledkem obou výše uvedených skutečností (preference nespolehlivé a nespojované komunikace na úrovni síťové vrstvy) je pak to, že TCP/IP nepotřebuje rozlišovat různé kategorie síťových přenosových protokolů, jako RM ISO/OSI (viz jeho kategorie A, B a C). Místo toho vystačí jen s jedním jediným síťovým protokolem, a to protokolem IP. Snad netřeba už tolik zdůrazňovat, že funguje nespojovaně a nespolehlivě - a že dokáže fungovat snad v jakémkoli prostředí (nad jakoukoli přenosovou technologií linkové vrstvy, resp. vrstvy síťového rozhraní).

## Představa síťových a transportních protokolů v TCP/IP

V TCP/IP pak nad takto koncipovanou síťovou vrstvou existuje transportní vrstva, která nabízí dva různé transportní protokoly (místo pěti v ISO/OSI). Liší se v tom, zda zachovávají celkový



způsob fungování síťového protokolu IP (tj. jeho nespojovaný a nespolehlivý charakter komunikace), nebo zda se jej snaží měnit - na spojovaný a spolehlivý. S oběma takto koncipovanými transportními protokoly jsme se už v předchozích dílech setkali:



**Rodina protokolů TCP/IP** obsahuje sadu protokolů pro komunikaci v počítačové síti a je hlavním protokolem celosvětové sítě Internet. Komunikační protokol je množina pravidel, které určují syntaxi a význam jednotlivých zpráv při komunikaci.

## IP (Internet Protocol)

Internet Protocol je základní protokol síťové vrstvy a celého Internetu. Provádí vysílání **datagramů** na základě síťových IP adres obsažených v jejich záhlaví. Poskytuje vyšším vrstvám **síťovou službu bez spojení**.

Každý datagram je samostatná datová jednotka, která obsahuje všechny potřebné údaje o adresátovi i odesílateli a pořadovém čísle datagramu ve zprávě. Datagramy putují sítí nezávisle na sobě a pořadí jejich doručení nemusí odpovídat pořadí ve zprávě. Doručení datagramu není zaručeno, spolehlivost musí zajistit vyšší vrstvy (TCP, aplikace).

Tento protokol se dále stará o segmentaci a opětovné sestavení datagramů do a z rámců podle protokolu nižší vrstvy (např. ethernet).

V současné době je převážně používán protokol IP verze 4. Je připravena nová verze 6, která řeší nedostatek adres v IPv4, bezpečnostní problémy a vylepšuje další vlastnosti protokolu IP.

### IPv4 - Internet protokol verze 4

- 32 bitové adresy
- cca 4 miliardy různých IP adres, dnes nedostačující

### IPv6 - Internet protokol verze 6

- 128 bitové adresy
- podpora bezpečnosti
- podpora pro mobilní zařízení
- funkce pro zajištění úrovně služeb (QoS - Quality of Service)
- fragmentace paketů - rozdělování
- jednoduchý přechod z IPv4 (musí podporovat systém, provider)
- snadnější automatická konfigurace (NDP - Neighbor discovery protocol)

## TCP (Transmission Control Protocol)

Transmission Control Protocol vytváří virtuální okruh mezi koncovými aplikacemi, tedy **spolehlivý přenos dat**. TCP je spojově orientovaný protokol. Zajišťuje:

- **proudový přenos dat** – není potvrzován každý paket, ale skupina (window); každý segment je identifikován sekvenčním číslem; pokud TCP dostane od vyšších vrstev balík dat, rozdělí je do segmentů, označí sekvenčním číslem a pošle 3. vrstvě (IP) k přenosu; Transparentní přenos libovolných dat.
- **spolehlivost** – zajištěna potvrzováním příjmu skupiny paketů; Spolehlivá transportní služba, doručí adresátovi všechna data bez ztráty a ve správném pořadí. **Služba se spojením, má fáze navázání spojení, přenos dat a ukončení spojení**.
- **ztracené nebo opožděné pakety příjemce nepotvrdí a odesílatel je pošle znovu**; efektivní řízení toku – příjemce nepotvrzuje každý paket, ale skupinu (příjemce informuje odesílatele jaké množství paketů je schopen přijmout čímž je zamezeno přetečení jeho interních bufferů);
- **plně duplexní operaci** – TCP umožňuje přijímat i odesílat data současně;
- **multiplexing** – možnost datových toků různých aplikací vyšších vrstev najednou prostřednictvím jednoho spojení. **Rozlišování aplikací pomocí portů**.

Aplikace používající TCP jako přenosový protokol – např. FTP, SMTP, http, SAP, SMB, ...



# UDP (User Datagram Protocol)

User Datagram Protocol poskytuje nespolehlivou transportní službu pro takové aplikace, které nepotřebují spolehlivost, jakou má protokol TCP. Nemá fázi navazování a ukončení spojení a už první segment UDP obsahuje aplikační data. UDP je používán aplikacemi jako je DHCP, TFTP, SNMP, DNS a BOOTP.

Protokol používá podobně jako TCP čísla portů pro identifikaci aplikačních protokolů.

- UDP je **nespojový** (connectionless) **protokol**;
- **nepřináší vlastnosti spolehlivosti přenosu, řízení toku nebo funkcí opravy chyb**;
- jde o jednoduchý interfejs mezi protokoly vyšší vrstvy a IP protokolem,
- hlavička protokolu UDP obsahu menší množství informací než hlavička TCP a tím má tento **protokol menší režii**;
- protože **UDP nemá žádné mechanismy pro dorozumívání se obou komunikujících stran** je přenos rychlejší než v případě TCP.

Aplikace používající UDP jako přenosový protokol – např. SNMP, NFS, TFTP, protokoly pro VoIP (např. RTP), ...

**Logika existence dvou transportních protokolů, které jsou vzájemně alternativní, je v tom že aplikace si mohou samy svobodně vybrat, který z nich chtějí používat.**

Například "klasické" počítačové aplikace, jako například přenos souborů, el. pošta atd., preferují zajištění spolehlivosti, a dávají přednost protokolu TCP. Naproti tomu novější multimediální aplikace dávají přednost spíše protokolu

UDP, protože ten se nezdržuje zajišťováním spolehlivosti, a dokáže tak přenášet data rovnoměrněji a s menší prodlevou (latencí), než protokol TCP.



## Protokoly servisní.

### ARP

**Address Resolution Protocol** se v počítačových sítích s IP protokolem používá k **získání ethernetové MAC adresy sousedního stroje z jeho IP adresy**. Používá se v situaci, kdy je třeba odeslat IP datagram na adresu ležící ve stejné podsíti jako odesílatel. Data se tedy mají poslat přímo adresátovi, u něhož však odesílatel zná pouze IP adresu. Pro odeslání prostřednictvím např. Ethernetu ale potřebuje znát cílovou ethernetovou adresu. Proto vysílající odešle **ARP dotaz (ARP request)** obsahující hledanou IP adresu a údaje o sobě (vlastní IP adresu a MAC adresu). Tento dotaz se posílá linkovým broadcastem – na MAC adresu identifikující všechny účastníky dané lokální sítě (v případě Ethernetu na ff:ff:ff:ff:ff:ff). ARP dotaz nepřekročí hranice dané podsítě, ale všechna k ní připojená zařízení dotaz obdrží a jako optimalizační krok si zapíše údaje o jeho odesílateli (IP adresu a odpovídající MAC adresu) do své ARP cache. Vlastník hledané IP adresy pak odešle tazateli **ARP odpověď (ARP reply)** obsahující vlastní IP adresu a MAC adresu. Tu si tazatel zapíše do **ARP cache** a může odeslat datagram.

**Pokud hledaný uzel není ve stejném segmentu, odpoví svou adresou příslušný směrovač.**

Informace o MAC adresách odpovídajících jednotlivým IP adresám se ukládají do **ARP cache**, kde jsou uloženy do vypršení své platnosti. Není tedy třeba hledat MAC adresu před odesláním každého datagramu – jednou

získaná informace se využívá opakovaně. V řadě operačních systémů ( Linux, Windows XP ) lze obsah ARP cache zobrazit a ovlivňovat příkazem **arp -a**.

Alternativou pro počítač bez ARP protokolu je používat tabulku přiřazení MAC adres IP adresám definovanou jiným způsobem, například pevně konfigurovanou. Tento přístup se používá především v prostředí se zvýšenými nároky na bezpečnost, protože v ARP se dá podvádět – místo skutečného vlastníka hledané IP adresy může odpovědět někdo jiný a stáhnout tak k sobě data pro něj určená.

ARP je definováno v doporučení RFC 826. Používá se pouze pro IPv4, novější verze IP protokolu, IPv6, používá podobný mechanismus nazvaný **Neighbour Discovery Protocol** (NDP, „objevování sousedů“).

Ačkoliv se ARP v praxi používá téměř výhradně pro překlad IP adres na MAC adresy, nebyl původně vytvořen pouze pro IP síť. ARP se může použít pro překlad MAC adres mnoha různých protokolů na síťové vrstvě. ARP byl také uzpůsoben tak, aby vyhodnocoval jiné typy adres fyzické vrstvy: například ATARP se používá k vyhodnocení ATM NSAP adres v protokolu **Classical IP over ATM**.

#### • ARP

- o známe cílovou IP adresu, zjišťujeme MAC, abychom mohli IP paket umístit do ethernetového rámce (linková vrstva) a odeslat
- o vytváří ARP tabulku, MAC↔IP
- o 1 MAC může mít více IP adres
- o vytváří dynamický záznam (dynamické záznamy mají omezenou životnost – sekundy až desítky minut)

#### • Proxy ARP

- o router se při ARP odpovědích vydává za PC, které leží za ním
- o vytvoření 1 logické podsítě z více fyzických sítí

ARP protokol se ale nedostane za hranice lokální sítě, k tomu slouží právě proxy ARP, která umí propojit dvě sítě, takže jejich zařízení spolu mohou komunikovat, jako by byly v jedné společné lokální síti.

**Princip Proxy ARP** - Koncové zařízení odešle ARP dotaz obsahující IP adresu zařízení, se kterým se chce spojit. ARP dotaz je rozeslán všem zařízením v dané lokální síti, ve které ovšem hledané zařízení není, ale je v ní zapojen router, na kterém je zapnut proxy ARP. Router předá dotaz do druhé sítě, ve které se nachází zařízení s hledanou IP adresou. Zároveň také místo cílového zařízení odpoví tazateli a pošle mu svoji MAC adresu. Tazatel tedy pak komunikuje s routerem, jako by komunikoval přímo s cílovým zařízením. Router pak dále komunikuje s cílovým zařízením a předává mu zprávy od tazatele. Jednotlivá zařízení v různých sítích tedy spolu komunikují, jako by byly v jedné lokální síti, ale ve skutečnosti veškerá jejich komunikace prochází přes router se zapnutou proxy ARP.

## RARP

**Reverse Address Resolution Protocol** se v počítačových sítích s IP protokolem používá:

- o když známe svoji MAC, ale neznáme svoji IP adresu
- o pro konfiguraci bezdiskových stanic → dnes **DHCP** (Dynamic Host Configuration Protocol)
- o umí předat jen IP → omezené možnosti
- o v síti je server se statickou ARP tabulkou
- o následník BootP, zpětně nekompatibilní
- o dynamické konfigurace stanic pro TCP/IP
- o server přiděluje:
  - a) staticky - dle MAC přidělí IP
  - b) dynamicky - stanice dostanou volnou IP z definovaného rozsahu
- o stanice si „pronájem“ obnovují podle požadovaného serveru → evidence aktivních stanic

- stanice požádá pomocí broadcastu (se svojí MAC), server odpoví

**Princip RARP** - Vysílající vyšle **RARP dotaz (RARP request)** obsahující vlastní MAC adresu. Dotaz se posílá na MAC broadcast, tedy všem počítačům v dané fyzické síti. V ní by se měl nacházet RARP server opatřený tabulkou obsahující IP adresy příslušející jednotlivým MAC adresám. Server prohlédne tabulku a pokud v ní najde MAC adresu tazatele, pošle mu zpět **RARP odpověď (RARP reply)** s IP adresou, kterou si má nastavit. RARP umožňuje centrální správu IP adres, trpí však dvěma významnými **nedostatků**:

1. Dotaz se posílá na fyzickou (MAC) broadcastovou adresu, nepřekročí tedy hranice fyzické sítě. V důsledku toho nelze mít v rozsáhlejší síti složené z několika podsítí jeden společný RARP server.
2. Předává pouze IP adresu. Stanice však ke svému síťovému životu potřebuje více informací (masku podsítě, implicitní bránu, adresu DNS serveru). Tyto informace nelze přenášet prostřednictvím RARP.

Důsledkem těchto nevýhod je, že se RARP prakticky nepoužívá. Pro automatickou konfiguraci stanic se častěji nasazují lepší protokoly DHCP nebo BOOTP. RARP je definováno v RFC 903.

## ICMP

**Internet Control Message Protocol** je **jeden z nejdůležitějších protokolů ze sady protokolů internetu**. Používají ho operační systémy počítačů v síti **k přenosu řídicích hlášení** (pro odesílání chybových zpráv, například pro oznámení, že požadovaná služba není dostupná nebo že potřebný počítač nebo router není dosažitelný).

ICMP se svým účelem liší od TCP a UDP protokolů tím, že se obvykle **nepoužívá** síťovými aplikacemi přímo.

Výjimkou je např. nástroj **ping**, který posílá ICMP zprávy „**Echo Request**“ (a očekává příjem zprávy „**Echo Reply**“), aby **určil, zda je cílový počítač dosažitelný a jak dlouho paketům trvá, než se dostanou k cíli a zpět**.

### Technický přehled

ICMP protokol definovaný v RFC 792 je součástí sady protokolů internetu. ICMP zprávy se typicky generují při chybách v IP datagramech (specifikováno v RFC 1122) nebo pro diagnostické a routovací účely.

Verze ICMP pro IPv4 je známá jako **ICMPv4**. IPv6 používá obdobný protokol **ICMPv6**.

ICMP zprávy se konstruují nad IP vrstvou; obvykle z IP datagramu, který ICMP reakci vyvolal. IP vrstva patřičnou ICMP zprávu zapouzdří novou IP hlavičkou (aby se ICMP zpráva dostala zpět k původnímu odesílateli) a obvyklým způsobem vzniklý datagram odešle.

Například každý stroj (jako třeba mezilehlé routery), který přeposílá IP datagram, musí v IP hlavičce dekrementovat políčko **TTL („time to live“, „zbývající doba života“)** o jedničku. **Jestliže TTL klesne na 0 (a datagram není určen stroji provádějícímu dekrementaci), router přijatý paket zahodí a původnímu odesílateli datagramu pošle ICMP zprávu „Time to live exceeded in transit“ („během přenosu vypršela doba života“).**

**Každá ICMP zpráva je zapouzdřená přímo v jediném IP datagramu, a tak (jako u UDP) ICMP nezaručuje doručení.**

Ačkoli ICMP zprávy jsou obsažené ve standardních IP datagramech, ICMP zprávy se zpracovávají odlišně od normálního zpracování protokolů nad IP. V mnoha případech je nutné prozkoumat obsah ICMP zprávy a doručit patřičnou chybovou zprávu aplikaci, která vyslala původní IP paket, který způsobil odeslání ICMP zprávy k původci.

Mnoho běžně používaných síťových diagnostických utilit je založeno na ICMP zprávách.

Příkaz **tracroute (Linux) tracert (Win)** je implementován odesláním UDP datagramů se speciálně nastavenou životností v TTL políčku IP hlavičky a očekáváním ICMP odezvy „**Time to live exceeded in transit**“ nebo „**Destination unreachable**“ („Cíl nedosažitelný“).

Utilita **ping** je implementována použitím ICMP zpráv „**Echo Request**“ a „**Echo Reply**“.

### Nejpoužívanější ICMP datagramy

- **Echo Request** ... požadavek na odpověď, každý prvek v síti pracující na IP vrstvě by na tuto výzvu měl reagovat. Často to z různých důvodů není dodržováno.
- **Echo Reply** ... odpověď na požadavek
- **Destination Unreachable** ... informace o nedostupnosti cíle, obsahuje další upřesňující informaci
  - **Net Unreachable** ... nedostupná cílová síť, reakce směrovače na požadavek komunikovat se sítí, do které nezná cestu
  - **Host Unreachable** ... nedostupný cílový stroj
  - **Protocol Unreachable** ... informace o nemožnosti použít vybraný protokol
  - **Port Unreachable** ... informace o nemožnosti připojit se na vybraný port
- **Redirect** ... přesměrování, používá se především pokud ze sítě vede k cíli lepší cesta než přes defaultní bránu. Stanice většinou nepoužívají směrovací protokoly a proto jsou informovány touto cestou. Funguje tak, že stanice pošle datagram své, většinou defaultní, bráně, ta jej přepošle správným směrem a zároveň informuje stanici o lepší cestě.
  - **Redirect Datagram for the Network** ... informuje o přesměrování datagramů do celé sítě
  - **Redirect Datagram for the Host** ... informuje o přesměrování datagramů pro jediný stroj
- **Time Exceeded** ... vypršel časový limit
  - **Time to Live exceeded in Transit** ... během přenosu došlo ke snížení TTL na 0, aniž byl datagram doručen
  - **Fragment Reassembly Time Exceeded** ... nepodařilo se sestavit jednotlivé fragmenty v časovém limitu (např. pokud dojde ke ztrátě části datagramů)

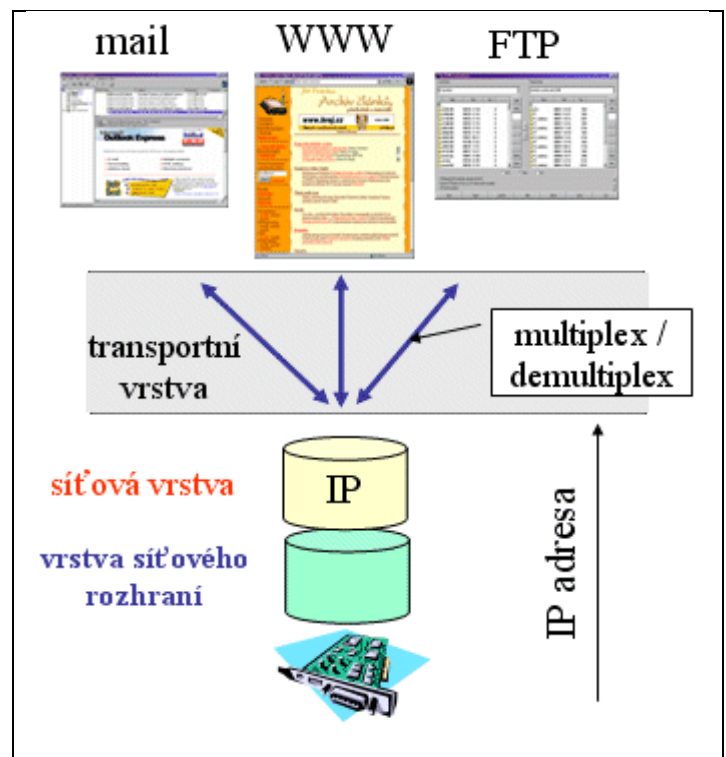
Ostatní datagramy jsou používány spíše vzácně, někdy je používání ICMP znemožněno špatným nastavením firewallu.

### Další úkoly transportní vrstvy

Transportní vrstva má obecně (v RM ISO/OSI i v TCP/IP) ještě další úkoly. Transportní vrstva je první vrstvou (měřeno "odspodu"), která se již nedívá na jednotlivé uzly jako na dále nedělitelné celky, ale rozlišuje v nich jednotlivé entity, které vystupují buď jako příjemci, nebo jako odesílatelé dat.

**O co vlastně jde:** na jednom počítači může běžet několik aplikací, které komunikují s dalšími uzly v síti, a to nezávisle na sobě. Například uživatel může mít na svém osobním počítači puštěný WWW browser, skrze který právě brouzdá báječným světem WWW. Vedle toho má puštěného i klienta elektronické pošty, skrze kterého přijímá svou elektronickou korespondenci. Tím výčet možností samozřejmě zdaleka nekončí, ale pro náš podstaty problému stačí alespoň dva různé běžící programy, resp. aplikace. Představme si totiž, co se

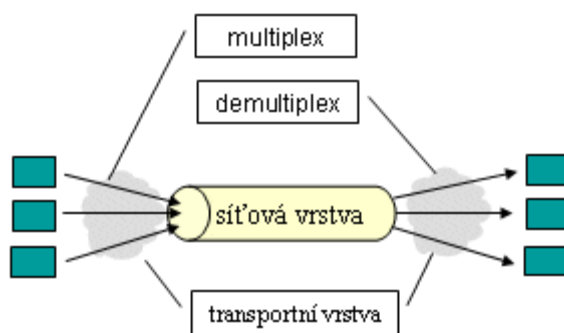
má stát v okamžiku, kdy uživatelův počítač přijme nějaká data. Komu mají být předána? Webovému browseru? Nebo emailovému klientovi? Jistě by to mělo záviset na tom, zda přijatá data představují nějakou WWW stránku, zprávu el. pošty, nebo třeba soubory, přenášené prostřednictvím protokolu FTP či něco ještě jiného



Síťová vrstva takovéto otázky neřeší. Pro ni je každý uzel sítě dále nedělitelný celek. Takže když přijme nějaká data pro daný uzel, nezajímá se o to, komu (které aplikaci) patří, a vždy je předá své bezprostředně vyšší vrstvě - vrstvě transportní. Teprve transportní vrstva pak má za úkol rozlišovat, komu - v rámci daného počítače - přijatá data patří, a předat je právě a pouze jemu. Obdobně při odesílání, kdy transportní vrstva přijímá data - zde také rozlišuje, od koho jsou (a tuto informaci musí vhodným způsobem zachovat pro koncového příjemce dat).

Můžeme si to celé představit také tak, že síťová vrstva vytváří přenosový kanál, který je společně využíván všemi odesílateli, resp. příjemci na daném uzlu, s tím že až transportní vrstva zajišťuje na straně odesílatele potřebné "sloučení" (multiplex) dat od různých odesílatelů, do jednoho datového toku, a na straně příjemce zase potřebné "rozbočení" (demultiplex) těchto dat podle toho, komu jsou určena. Vše naznačuje i následující obrázek, který je oproti obvyklým konvencím jakoby pootočen o 90 stupňů.

Jinou představu téhož ukazuje i další obrázek, na kterém již lze názorně ukázat jeden významný aspekt, týkající se adresování. Je opět společný pro RM ISO/OSI i TCP/IP, a týká se adresování na úrovni transportní vrstvy.



## Aplikační protokoly (služby) s vazbou na TCP/UDP.

- **DNS** – systém doménových jmen
- **DHCP** – dynamické přidělování IP adres
- **FTP** – přenos souborů po síti
- **TFTP** - jednoduchý protokol pro přenos souborů
- **HTTP** – přenos hypertextových dokumentů (WWW)
- **WebDAV** – rozšíření HTTP o práci se soubory
- **IMAP** (Internet Message Access Protocol) umožňuje manipulovat s jednotlivými e-mailovými zprávami na poštovním serveru.
- **IRC** (Internet Relay Chat) – jednoduchý chat po internetu.
- **NNTP** (Network News Transfer Protocol) umožňuje číst a umísťovat do sítě zprávy typu *news*.
- **NFS** (Network File System) – síťový systém souborů, který umožňuje transparentní sdílení vzdálených souborů jakoby byly lokální.
- **NTLM** Autentizační protokol Windows
- **NTP** – synchronizace času (šíření přesného času)
- **POP3** (Post Office Protocol) – protokol pro získání pošty z poštovního serveru.
- **SMB** (Server Message Block) - sdílení souborů a tiskáren v sítích Windows
- **SMTP** – zasílání elektronické pošty
- **SNMP** Simple Network Management Protokol je určen pro správu síťových uzlů.
- **Telnet** – protokol virtuálního terminálu.
- **SSH** – bezpečný shell
- **X11** – zobrazování oken grafických programů v Unixových systémech
- **XMPP** – rozšiřitelný protokol pro zasílání zpráv a sledování přítomnosti (protokol Jabber)

## Vybrané aplikační protokoly:

### TFTP

**Trivial File Transfer Protocol (TFTP)** je velice jednoduchý protokol pro přenos souborů, obsahující jen základní funkce protokolu FTP. Jeho specifikace byla poprvé uveřejněna v roce 1980.

TFTP je určen pro přenos souborů v případech, kdy je běžný protokol FTP nevhodný pro svou komplikovanost. Typickým případem je bootování bezdiskových počítačů ze sítě (viz také BOOTP), kdy se celý přenosový protokol musí vejít do omezeného množství paměti, která je k dispozici na bezdiskovém stroji.

TFTP je založen na části protokolu EFTP, což je část univerzálního balíku PARC.

Původní verze TFTP před RFC 1350 obsahovala závažnou chybu (tzv. *Sorcerer's apprentice syndrome*), která za jistých okolností způsobila množení dat přenášených po síti.

#### Technické informace

Jelikož TFTP funguje nad nespojovaným protokolem UDP, musí obsahovat vlastní řízení spojení. Koncepce sezení je jednoduchá: v jednom spojení lze přenést jen jediný soubor, při komunikaci se na síti pohybuje vždy jen jediný paket (po odeslání jednoho paketu program čeká na jeho potvrzení a teprve poté posílá další).

Kvůli tomuto zjednodušení poskytuje protokol na linkách s velkou latencí jen malou přenosovou rychlost.

TFTP používá portu 69 (FTP používá spojovaný protokol TCP a port 21).

Oproti FTP má různá omezení a odlišnosti:

- Nelze procházet adresáře.
- Neumožňuje přihlášení uživatele ani zadání hesla.
- Je používán pro čtení nebo zápis dat na vzdálený server.
- Podporuje tři odlišné přenosové módy: **netascii** (pro text v ASCII s úpravami z protokolu Telnet), **octet** (pro syrová binární 8bitová data) a **mail** (pro zaslání e-mailové zprávy; tento mód by se už neměl používat).
- Maximální velikost přenášeného souboru je 32 MB.

Kvůli nedostatečnému zabezpečení je nebezpečné používat tento protokol k výměně dat přes internet, používá se výhradně v lokálních sítích, kde nehrozí takové nebezpečí zcizení nebo poškození dat.

Další vylepšení byly později implementovány do RFC 2347, ale protokol zůstává zpětně kompatibilní.

#### Detaily TFTP

- Klient si při vytváření spojení zvolí číslo spojení (*transmit identifikátor* – TID). Toto číslo je stále během celé doby trvání spojení a toto spojení identifikuje (neboť použitý protokol UDP je bezstavový). TID se může volit náhodně, pravděpodobnost, že dva klienti vyberou v jednu dobu dvě shodná čísla je velmi malá. Každému paketu jsou přiřazena dvě TID (zdrojové a cílové). Tato TID se v UDP protokolu (nebo v jiném datagramovém protokolu) použijí jako zdrojový a cílový port.
- Klient si zvolí svoje zdrojové TID a zašle ho s požadavkem na standardní TID (69 dekadicky) serveru. Tudíž uživatel A pošle z portu X paket RRQ (žádost o čtení) nebo WRQ (žádost o zápis) uživateli (serveru) B na port 69 (obvykle). Tento paket obsahuje jméno souboru, cestu a použitý přenosový mód.
- Po zaslání požadavku musí následovat kladná odpověď serveru B z portu Y na port X uživatele A, kterou může být buď potvrzovací paket (při zápisu), nebo první datový paket (při čtení). Posílá se maximálně 512 bajtů dat v jednom paketu, kratší paket znamená konec souboru.

#### Syntaxe příkazu tftp pro přenos souboru do a z počítače přes TFTP.

TFTP [-i] cíl [GET | PUT] zdroj [umístění]



## TELNET a SSH

**Telnet** (zkratka z *Telecommunication Network*) je v informatice označení protokolu používaném v počítačových sítích, který pomocí stejnojmenné aplikace umožňuje uživateli připojení ke vzdálenému počítači.

### Protokol telnet

Pracuje na aplikační vrstvě používaný TCP/IP. Používá se v Internetu pro realizaci spojení typu **klient-server** protokolem **TCP**, přičemž přenáší osmibitové znaky oběma směry (duplexní spojení). **Serverová část standardně naslouchá na portu číslo 23**. Součástí protokolu je vyjednávání nastavení určitých voleb důležitých pro vzájemnou komunikaci.

### Program TELNET

TELNET je též označení pro program, který realizoval komunikaci mezi dvěma počítači pomocí telnet protokolu. Program je dodnes součástí Microsoft Windows a unixových systémů. Dříve se používal (spolu s protokolem telnet) pro připojení ke vzdálenému počítači prostřednictvím počítačové sítě jako emulace terminálu, která poskytovala možnost práce uživatele na vzdáleném počítači v příkazovém řádku. Telnet tak byl nástupcem terminálů, ze kterých se uživatelé připojovali ke vzdálenému počítači pomocí sériové linky. Hlavní nevýhodou telnetu je absence šifrování přenášených dat, a proto dnes uživatelé místo telnetu používají protokol SSH. V současné době se program telnet používá pro ruční komunikaci mezi počítačovými programy (například simulace připojení webového prohlížeče k webovému serveru, při simulaci SMTP protokolu pro přepravu elektronické pošty a podobně).

### Základní služby Telnetu

V dokumentu RFC 854 jsou definovány tři základní služby:

- **Síťový virtuální terminál** (NVT – Network Virtual Terminal), který poskytuje standardní rozhraní.
- **Vyjednávání klienta/serveru** o nastavení určitých voleb
- **Symetrické zobrazení** terminálu a procesů.

Síťový virtuální terminál zajišťuje průhlednost všech operací vůči uživateli. Nejsou zde rozdíly mezi jednotlivými komunikujícími zařízeními. Virtuální terminál poskytuje obecnou sadu příkazů pro všechny typy zařízení. Pro přenos využívá spolehlivé přenosové služby **TCP**, ale jen **poloduplexním způsobem**.

Definovaný formát NVT používá sedmibitový kód ASCII pro znaky a zobrazení. Díky tomu může Telnet operovat na různých operačních systémech. Klávesnice NVT generuje všech 128 kódů ASCII pomocí kláves, klávesových kombinací.

### Příkazy Telnetu vyjednávání o volbách

Přestože obě komunikující strany předpokládají, že je protějščí strana vybavena NVT, proběhne nejdříve výměna údajů, ve které se obě strany dohodnou na určitých parametrech a volbách komunikace. Pokud jedna strana neumí použít danou volbu, požadavek zamítne. Strany však musí dodržet minimální standard NVT.

### Existují čtyři možné požadavky:

- **WILL** - odesílatel chce danou volbu zapnout
- **DO** - odesílatel chce, aby příjemce danou volbu zapnul
- **WONT** - odesílatel chce danou volbu vypnout
- **DONT** - odesílatel chce, aby příjemce danou volbu vypnul

Lidé si občas pletou protokol telnet s programem telnet, který dokáže sloužit jako klient pro připojení na telnetový nebo jiný server používající plaintext TCP protokol.

### Bezpečnost

Když byl TELNET poprvé vyvinut v roce 1969, většina uživatelů počítačové sítě byla buď na univerzitách, ve vládě nebo ve velkých společnostech. Za této skutečnosti nebyla bezpečnost brána tak vážně jako po celosvětovém rozmachu internetu. Jak se zvyšoval počet uživatelů internetu, rostl také počet těch, kteří se zkusili nabourat do serverů ostatních. Tím vzrostla potřeba šifrování.

Experti na počítačovou bezpečnost již nadále nedoporučovali použití TELNETu pro vzdálený přístup k serverům pro následující důvody: **TELNET standardně nešifroval žádná data odesílaná do internetu** (včetně hesel), často se také **dala komunikace přímo odposlouchávat** a dále takto odposlechnutá hesla zneužít; kdokoliv kdo měl přístup do routeru, switchu nebo hubu umístěného na síti mezi dvěma zařízeními, kde byl TELNET používán, mohl zachytit celou jejich komunikaci a získat tak hesla a vše co bylo napsáno použitím několika běžných programů jako **Tcpdump a Wireshark**. **Většina implementací TELNETu postrádala jakékoliv ověřování**, které by bylo schopno zajistit, aby nemohlo docházet k odposlouchávání. Většina běžně užívaných klientů telnetu měla několik zranitelných míst zjištěných v průběhu let.

Tyto bezpečnostní nedostatky způsobily rapidní **opouštění od použití TELNETu, zvláště pak v internetu ve prospěch protokolu SSH**, který byl poprvé uvolněn v roce 1995. **SSH poskytovalo většinu funkcí TELNETu a navíc silné šifrování**, které bránilo v získávání hesel a veřejný klíč, který sloužil k ověření vzdáleného počítače. Jak se stalo u mnoha původních internetových protokolů, vznikla rozšíření, která telnetu dala TLS bezpečnost a SASL ověřování a snažila se opravit původní nedostatky. Stále však byly části protokolu, které tyto rozšíření nepodporovaly.

### Využití pro ladění

Klientským programem TELNETu se lze pro testovací účely připojit na jinou textově orientovanou službu Internetu. Lze tak **například simulovat činnost webového prohlížeče** – zadáním příkazu (pozor na velikost písmen):

**telnet cs.wikipedia.org 80**

**GET /http /1.0**

**Host: cs.wikipedia.org**

stisknutím 2× Enter se pomocí HTTP protokolu provede nejprve připojení k webovému serveru a pak žádost o zobrazení hlavní stránky Wikipedie. Podobným jednoduchým postupem můžeme snadno v případě problémů zjistit, zda nějaká služba vůbec běží a zda alespoň v základních rysech funguje.

## DNS

Domain Name System je **hierarchický systém doménových jmen**, který **je realizován servery DNS** a protokolem stejného jména, kterým si vyměňují informace. **Jeho hlavním úkolem a příčinou vzniku jsou vzájemné převody doménových jmen a IP adres uzlů sítě**. Později ale přibral další funkce (např. pro elektronickou poštu či IP telefonii) a slouží dnes de facto jako distribuovaná databáze síťových informací.

**Protokol používá porty TCP/53 i UDP/53**, je definován v RFC1035. **Servery DNS jsou organizovány hierarchicky, stejně jako jsou hierarchicky tvořeny názvy domén**. Jména domén umožňují lepší orientaci lidem, adresy pro stroje jsou však vyjádřeny pomocí adres **32bitových (IPv4) A záznam** nebo **128bitových (IPv6) - AAAA záznam**. **Systém DNS umožňuje efektivně udržovat decentralizované databáze doménových jmen a jejich překlad na IP adresy. Stejně tak zajišťuje zpětný překlad IP adresy na doménové jméno - PTR záznam (Pointer Record).**

Protože je používání názvů pro člověka daleko příjemnější než používání číselných adres, vznikla už v dobách ARPAnetu potřeba takový převod realizovat. Původně byl na všechny počítače distribuován jediný soubor (v Unixu **/etc/hosts**). Tato koncepce přestala velmi rychle vyhovovat potřebám a především nárokům na rychlou aktualizaci. Přesto se tento soubor dodnes používá, v závislosti na konfiguraci systému je možné jej použít buď prioritně před dotazem na DNS nebo v případě, že DNS neodpovídá. Je možné jej také použít k uložení vlastních přezdívek pro často navštěvované servery, případně také pro blokování reklam a podobně.

### Jak DNS funguje.

**Prostor doménových jmen tvoří strom**. Každý uzel tohoto stromu obsahuje informace o části jména (doméně), které je mu přiděleno a odkazy na své podřízené domény. Kořenem stromu je tzv. **kořenová domé-**

na, která se zapisuje jako samotná tečka. Pod ní se v hierarchii nacházejí tzv. **domény nejvyšší úrovně** (*Top-Level Domain, TLD*). Ty jsou buď **tematické** (**com** pro komerci, **edu** pro vzdělávací instituce atd.) nebo **státní** (**cz** pro Česko, **sk** pro Slovensko, **jo** pro Jordánsko atd.).

Strom lze administrativně rozdělit do zón, které spravují jednotliví správci (organizace nebo i soukromé osoby), přičemž taková zóna obsahuje autoritativní informace o spravovaných doménách. Tyto informace jsou poskytovány autoritativním DNS serverem.

Výhoda tohoto uspořádání spočívá v možnosti zónu rozdělit a správu její části svěřit někomu dalšímu. Nově vzniklá zóna se tak stane autoritativní pro přidělený jmenný prostor. Právě možnost delegování pravomocí a distribuovaná správa tvoří klíčové vlastnosti DNS a jsou velmi podstatné pro jeho úspěch. Ve vyšších patrech doménové hierarchie platí, že zóna typicky obsahuje jednu doménu. Koncové zóny přidělené organizacím připojeným k Internetu pak někdy obsahují několik domén – například doména *kdesi.cz* a její poddomény *vyroba.kdesi.cz*, *marketing.kdesi.cz* a *obchod.kdesi.cz* mohou být obsaženy v jedné zóně a obhospodařovány stejným serverem.

### Složení doménového jména

Celé jméno se skládá z několika částí oddělených tečkami. Na jeho konci se nacházejí domény nejobecnější, směrem doleva se postupně konkretizuje.

- část nejvíce vpravo je doména nejvyšší úrovně, např. *wikipedia.org* má TLD *org*.
- jednotlivé části (subdomény) mohou mít až 63 znaků a skládat se mohou až do celkové délky doménového jména 255 znaků. Doména může mít až 127 úrovní. Bohužel některé implementace jsou omezeny více.

### DNS servery (name servery)

DNS server může hrát vůči doméně (přesněji zóně, ale ve většině případů jsou tyto pojmy zaměnitelné) jednu ze tří rolí:

- **Primární server** je ten, na němž data vznikají. Pokud je třeba provést v doméně změnu, musí se editovat data na jejím primárním serveru. **Každá doména má právě jeden primární server.**
- **Sekundární server** je automatickou kopií primárního. Průběžně si aktualizuje data a slouží jednak jako záloha pro případ výpadku primárního serveru, jednak pro rozkládání zátěže u frekventovaných domén. Každá doména musí mít alespoň jeden sekundární server.
- **Pomocný (caching only) server** slouží jako vyrovnávací paměť pro snížení zátěže celého systému. Uchovává si odpovědi a poskytuje je při opakování dotazů, dokud nevyprší jejich životnost.

Odpověď pocházející přímo od primárního či sekundárního serveru je **autoritativní**, čili je brána za správnou. Z hlediska věrohodnosti odpovědi není mezi primárním a sekundárním serverem rozdíl, oba jsou autoritativní. Naproti tomu odpověď poskytnutá z vyrovnávací paměti není autoritativní. Klient může požádat o autoritativní odpověď, v běžných případech ale stačí jakákoli.

### Root servery

**Kořenové jmenné servery** (*root name servers*) představují zásadní část technické infrastruktury Internetu, na které závisí spolehlivost, správnost a bezpečnost operací na internetu. Tyto servery poskytují kořenový zónový soubor (*root zone file*) ostatním DNS serverům. Jsou součástí DNS, celosvětově distribuované databáze, která slouží k překladu unikátních doménových jmen na ostatní identifikátory.

Kořenový zónový soubor popisuje, kde se nacházejí autoritativní servery pro domény nejvyšší úrovně. Tento kořenový zónový soubor je relativně velmi malý a často se nemění – operátoři root serverů ho pouze zpřístupňují, samotný soubor je vytvářen a měněn organizací IANA.

Pojem root server je všeobecně používán pro 13 kořenových jmenných serverů. Root servery se nacházejí ve 34 zemích světa, na více než 80 místech. Root servery jsou spravovány organizacemi, které vybírá IANA. Následující tabulka zobrazuje těchto 13 root serverů:

Název root serveru	Operátor
<b>A</b>	VeriSign Global Registry Services
<b>B</b>	University of Southern California - Information Sciences Institute
<b>C</b>	Cogent Communications
<b>D</b>	University of Maryland
<b>E</b>	NASA Ames Research Center
<b>F</b>	Internet Systems Consortium, Inc.
<b>G</b>	U.S. DOD Network Information Center
<b>H</b>	U.S. Army Research Lab
<b>I</b>	Autonomica/NORDUnet
<b>J</b>	VeriSign Global Registry Services
<b>K</b>	RIPE NCC
<b>L</b>	ICANN
<b>M</b>	WIDE Proje

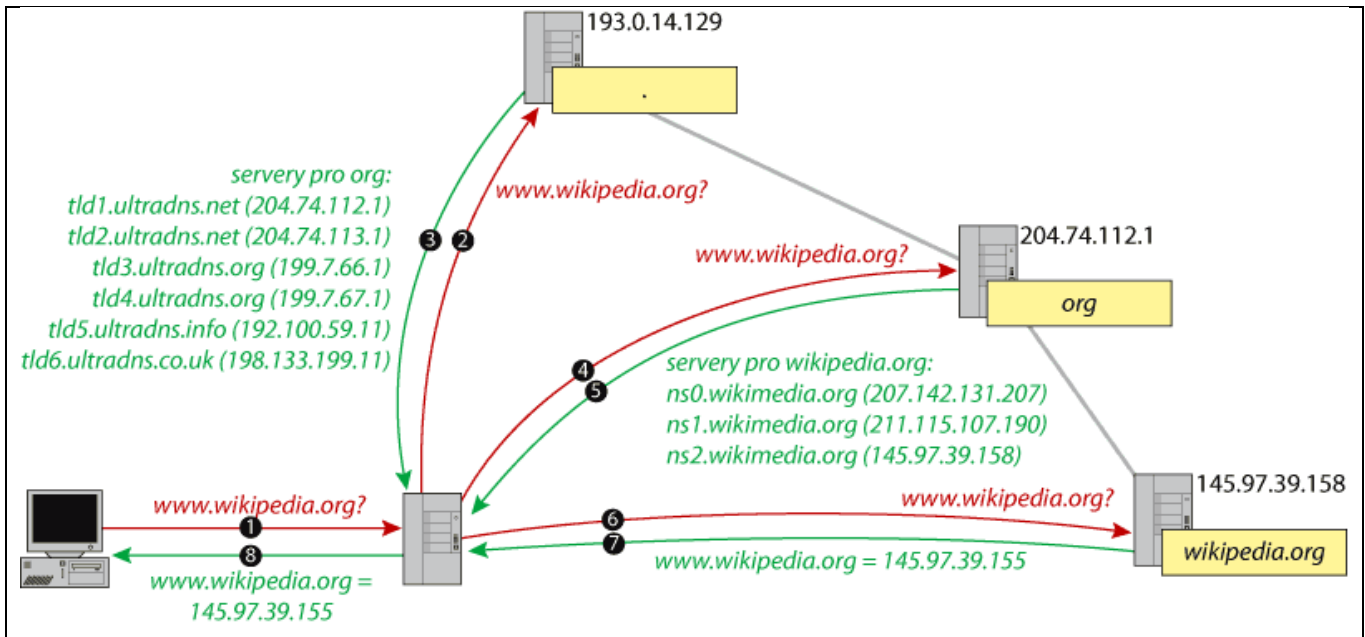
Více informací o umístění root serverů naleznete na oficiálních stránkách DNS root servers.

### Řešení dotazu

Každý koncový počítač má ve své konfiguraci síťových parametrů obsaženu i adresu lokálního DNS serveru, na nějž se má obracet s dotazy. V operačních systémech odvozených od Unixu je obsažena v souboru */etc/resolv.conf*, v MS Windows ji najdete ve vlastnostech protokolu TCP/IP (případně můžete z příkazového řádku v XP zadat textový příkaz *ipconfig /all*). Adresu lokálního serveru počítač typicky obdrží prostřednictvím DHCP. Pokud počítač hledá určitou informaci v DNS (např. IP adresu k danému jménu), obrátí se s dotazem na tento lokální server. Každý DNS server má ve své konfiguraci uvedeny IP adresy kořenových serverů (autoritativních serverů pro kořenovou doménu). Obrátí se tedy s dotazem na některý z nich.

Kořenové servery mají autoritativní informace o kořenové doméně. Konkrétně znají všechny existující domény nejvyšší úrovně a jejich autoritativní servery. Dotaz je tedy následně směrován na některý z autoritativních serverů domény nejvyšší úrovně, v níž se nachází cílové jméno. Ten je opět schopen poskytnout informace o své doméně a posunout řešení o jedno patro dolů v doménovém stromě. Tímto způsobem řešení postupuje po jednotlivých patrech doménové hierarchie směrem k cíli, až se dostane k serveru autoritativnímu pro hledané jméno, který pošle definitivní odpověď. Získávání informací z takového systému probíhá rekurzí. Resolver (program zajišťující překlad) postupuje od kořene postupně stromem směrem dolů dokud nenalezne autoritativní záznam o hledané doméně. Jednotlivé DNS servery jej postupně odkazují na autoritativní DNS pro jednotlivé části jména.

**Příklad:** Hledání IP adresy k adrese *www.wikipedia.org*:



#### Postup hledání *www.wikipedia.org*

1. Uživatel zadal do svého WWW klienta doménové jméno *www.wikipedia.org*. Resolver v počítači se obrátil na lokální DNS server s dotazem na IP adresu pro *www.wikipedia.org*.
2. Lokální DNS server tuto informaci nezná. Má však k dispozici adresy kořenových serverů. Na jeden z nich se obrátí (řekněme na 193.0.14.129) a dotaz mu přepoše.
3. Kořenový server také nezná odpověď. Ví však, že existuje doména nejvyšší úrovně *org*, a jaké jsou její autoritativní servery, jejichž adresy tazateli poskytne.
4. Lokální server jeden z nich vybere (řekněme, že zvolí *tld1.ultradns.net* s IP adresou 204.74.112.1) a pošle mu dotaz na IP adresu ke jménu *www.wikipedia.org*.
5. Oslovený server informaci opět nezná, ale poskytne IP adresy autoritativních serverů pro doménu *wikipedia.org*. Jsou to *ns0.wikimedia.org* (207.142.131.207), *ns1.wikimedia.org* (211.115.107.190) a *ns2.wikimedia.org* (145.97.39.158).
6. Lokální server opět jeden z nich vybere a pošle mu dotaz na IP adresu ke jménu *www.wikipedia.org*.
7. Jelikož toto jméno se již nachází v doméně *wikipedia.org*, dostane od jejího serveru nepochybně autoritativní odpověď, že hledaná IP adresa zní 145.97.39.155.
8. Lokální DNS server tuto odpověď předá uživatelskému počítači, který se na ni ptal.

Výše popsany postup popisuje kompletní řešení daného dotazu. Může se ale stát, že některý z oslovených serverů má hledanou informaci ve své vyrovnávací paměti, protože odpovídající dotaz nedávno řešil. V takovém případě poskytne neautoritativní odpověď z vyrovnávací paměti a další dotazování odpadá. Ve vyrovnávací paměti mohou být i mezivýsledky - například lokální DNS server v ní skoro jistě bude mít informaci o autoritativních serverech pro doménu *org*, protože v ní pravděpodobně hledá každou chvíli. V takovém případě by vypadly kroky 2 a 3 a lokální server by se s dotazem rovnou obrátil na některý z autoritativních serverů domény *org*.

**Oslovené servery v popsaném příkladu vykazují dva odlišné druhy chování.**

1. Při **rekurzivním řešení dotazu** se server chopí vyřízení dotazu, najde odpověď a pošle ji tazateli. Rekurzivní přístup server zatěžuje (musí sledovat postup řešení, ukládat si mezivýsledky apod.), ale projde jím odpověď a tu si může uložit do vyrovnávací paměti. Typicky se tak chovají **lokální servery**, aby si plnily vyrovnávací paměti a mohly dalším tazatelům poskytovat odpovědi rovnou.
2. Při **nerekurzivním řešení dotazu** server dotaz neřeší, pouze poskytne tazateli adresy dalších serverů, jichž se má ptát dál. Takto se chovají servery ve vyšších patrech doménové hierarchie (**kořenové a autorita-**

ativní servery TLD)), které by rekurzivní chování kapacitně nezvládaly. V příkladu výše se rekurzivně choval lokální server, zatímco autoritativní servery pro kořenovou doménu a doménu *org* se chovaly nerekurzivně (což odpovídá realitě).

### Reverzní dotazy

Nejběžnějším úkolem DNS je poskytnout informace (nejčastěji IP adresu) pro zadané doménové jméno. Dovede ale i opak – sdělit jméno, pod kterým je daná IP adresa zaregistrována. Při vkládání dat pro zpětné dotazy bylo ale třeba vyřešit problém s opačným uspořádáním IP adresy a doménového jména. Zatímco IP adresa má na začátku obecné informace (adresu sítě), které se směrem doprava zpřesňují až k adrese počítače, doménové jméno má pořadí přesně opačné. Instituce připojená k Internetu typicky má přidělen začátek svých IP adres a konec svých doménových jmen.

Tento nesoulad řeší DNS tak, že při reverzních dotazech obrací pořadí bajtů v adrese. K obrácené IP adrese pak připojí doménu *in-addr.arpa* a výsledné „jméno“ pak vyhledává standardním postupem. Hledá-li například jméno k IP adrese 145.97.39.155, vytvoří dotaz na *155.39.97.145.in-addr.arpa*. Obrácení IP adresy umožňuje delegovat správu reverzních domén odpovídajících sítím a podsítím správcům dotyčných sítí a podsítí. V příkladu použitou síť 145.97.0.0/16 spravuje nizozemský SURFnet a ten má také ve správě jí odpovídající doménu *97.145.in-addr.arpa*. Kdykoli zavede do sítě nový počítač, může zároveň upravit data v reverzní doméně, aby odpovídala skutečné situaci.

Je dobré mít na paměti, že na data z reverzních domén nelze zcela spoléhat. Do reverzní domény se v principu dají zapsat téměř libovolná jména. Nikdo například nemůže zabránit SURFnetu, aby o počítači 145.97.1.1 prohlásil v reverzní zóně, že se jedná třeba o *www.seznam.cz*. Pokud na tom záleží, je záhodno si poskytnutou informaci ověřit normálním dotazem (zde nalézt IP adresu k *www.seznam.cz* a porovnat ji s 145.97.1.1). Jestliže odpověď na něj bude původní IP adresa, jsou data důvěryhodná – správce klasické i reverzní domény tvrdí totéž. Pokud se liší, znamená to, že data v reverzní doméně jsou nekorektní.

### Alternativní stromy

Běžní uživatelé internetu se setkají jen s „oficiálním“ stromem domén. Je ovšem možné založit libovolné množství takových stromů, které mohou obsahovat i stejná jména. Příkladem budiž například neoficiální strom **czf** fungující uvnitř některých privátních sítí v ČR.

### DNS cache

Téměř každý DNS server funguje zároveň jako DNS cache. Při opakovaných dotazech pak nedochází k rekurzivnímu prohledávání stromu, ale odpověď je získána lokálně. V DNS záznamech je totiž uložena i informace jak dlouho lze záznam používat (TTL) a lze také zjistit, zda byl záznam změněn. Po vypršení platnosti je záznam z DNS cache odstraněn.

### Doba uložení záznamu

Problém s uložením záznamů v cache nastává v případě změny záznamu. Pokud administrátor nastaví TTL na 6 hodin, a poté provede změnu záznamu, nastane situace, že někteří uživatelé sítě dostanou informaci již novou a někteří ještě starou. Tato situace bude trvat právě oněch 6 hodin, v závislosti na nastavení ostatních serverů a také v závislosti na době, která uplynula od jejich posledního dotazu.

Je proto nutné zvolit správný poměr mezi rychlostí šíření změn a ušetřeným výkonem a přenosovým pásmem DNS serveru. Pokud se změny provádí často, je vhodné zvolit kratší TTL v řádu jednotek hodin, pokud se změny téměř neprovádějí, může být TTL ve dnech.



## Zónové soubory

Obsah zóny (domény či několika domén) je uložen v tzv. **zónovém souboru**. Skládá se z jednotlivých záznamů (přesný název zní **zdrojové záznamy**, resource records, RR) obsahujících dílčí informace. Jejich názvy a nesené informace jsou přesně definovány v příslušných dokumentech, většina v RFC 1035. Formát textového zápisu zónového souboru se liší v závislosti na použitém serveru, zde použijeme nejrozšířenější BIND.

Záznamy v něm mají tvar

### **jméno životnost třída typ parametry**

- **jméno** je doménové jméno, pro něž záznam vytváříte. Zpravidla patří do aktuálně definované domény, pak se píše jen samotné jméno bez tečky na konci a bude k němu doplněna aktuální doména. Pokud jméno ukončíte tečkou, nic se k němu nedoplňuje a bere se jako kompletní. Nemusí být uvedeno, pak se přebírá z předchozího záznamu.
- **životnost** určuje dobu platnosti záznamu v sekundách. Většinou se neuvádí a záznamům se ponechává implicitní životnost. Umožňuje vám však udělat výjimku.
- **třída** určuje rodinu protokolů, k níž se záznam vztahuje. DNS lze teoreticky používat i pro jiné síťové architektury, v praxi však třída vždy bývá **IN**, což znamená Internet.
- **typ** určuje typ definovaného záznamu (viz níže).
- **parametry** se vztahují k typu záznamu, poskytují mu potřebná data. Obsahem parametrů často bývají doménová jména. Je třeba zdůraznit, že v parametrech se smí vyskytovat jen skutečná jména, nikoli přezdívký zavedené pomocí **CNAME** (viz níže).

Zónový soubor vždy musí začínat záznamem typu SOA.

## Typy záznamů

Nejčastěji používané jsou následující typy zdrojových záznamů (v abecedním pořadí):

- **A** (address record) obsahuje IPv4 adresu přiřazenou danému jménu, například když jménu *cosi.kdesi.cz* náleží IP adresa 1.2.3.4, bude zónový soubor pro doménu *kdesi.cz* obsahovat záznam  
cosi IN A 1.2.3.4
- **AAAA** (IPv6 address record) obsahuje IPv6 adresu. Zmíněnému stroji bychom IPv6 adresu 2001:718:1c01:1:02e0:7dff:fe96:daa8 přiřadili záznamem  
cosi IN AAAA 2001:718:1c01:1:02e0:7dff:fe96:daa8
- **CNAME** (canonical **name** record) je **alias** - jiné jméno pro jméno již zavedené. Typicky se používá pro servery známých služeb, jako je například WWW. Jeho definice pomocí přezdívký umožňuje jej později snadno přestěhovat na jiný počítač. Pokud náš *cosi.kdesi.cz* má sloužit zároveň jako *www.kdesi.cz*, vložíme do zónového souboru  
www IN CNAME cosi
- **MX** (mail **exchange** record) **oznamuje adresu a prioritu serveru pro příjem elektronické pošty** pro danou doménu. Tentokrát jsou parametry dva - priorita (přirozené číslo, menší znamená vyšší prioritu) a doménové jméno serveru. Pokud poštu pro počítač *cosi.kdesi.cz* přijímá nejlépe počítač *mail.kdesi.cz* a případně jako záložní i *mail.jinde.cz*, bude zónový soubor obsahovat záznamy (všimněte si použití jmen s tečkou a bez tečky)  
cosi IN MX 10 mail  
IN MX 20 mail.jinde.cz.
- **NS** (name **server** record) ohlašuje **jméno autoritativního DNS serveru pro danou doménu**. Bude-li mít doména *kdesi.cz* poddoménu *obchod.kdesi.cz*, jejímiž servery budou *ns.kdesi.cz* (primární) a *ns.jinde.cz* (sekundární), bude zónový soubor pro *kdesi.cz* obsahovat  
obchod IN NS ns  
IN NS ns.jinde.cz.

- **PTR** (pointer record) je speciální typ záznamu pro reverzní zóny. Obsahuje na pravé straně jméno počítače přidělené adrese na straně levé (adresa je transformována na doménu výše popsáním postupem). Držme se našeho příkladu pro záznam typu A - v souladu s ním by zónový soubor pro doménu *3.2.1.in-addr.arpa* měl obsahovat (zónový soubor definuje reverzní doménu, proto je třeba psát na pravé straně kompletní jméno s tečkou, jinak by za ně připojil reverzní doménu)

4 IN PTR cosi.kdesi.cz.

- **SOA** (start of authority record) je zahajující záznam zónového souboru. Obsahuje jméno primárního serveru, adresu elektronické pošty jejího správce (zavináč je v ní ale nahrazen tečkou) a následující údaje:
  - *Serial* — sériové číslo, které je třeba zvětšit s každou změnou v záznamu. Podle něj sekundární server pozná, že v doméně došlo ke změně. Pokud jej zapomenete zvětšit, rozejde se obsah sekundárních serverů s primárním, což rozhodně není dobré. Pro přehlednost často ve formátu YYYYMMDDHH.
  - *Refresh* — jak často se má sekundární server dotazovat na novou verzi zóny (v sekundách).
  - *Retry* — v jakých intervalech má sekundární server opakovat své pokusy, pokud se mu nedaří spojit s primárním.
  - *Expire* — čas po kterém označí sekundární server své záznamy za neaktuální, pokud se jim nedaří kontaktovat primární server.
  - **TTL** — implicitní doba platnosti záznamů.

Časové údaje jsou v sekundách. Novější implementace umožní pro vyšší pohodlí používat k číslům přípony 'M','H','D' a 'W' (minuta, hodina, den, týden) – například 8h znamená 8 hodin, čili totéž co hodnota 28800. Podívejme se na příklad SOA záznamu:

@ IN SOA ns.kdesi.cz. franta.kdesi.cz. (200605140 1h 5m 1w 1d)

### Příklad zónového souboru

Zkusme dát dohromady lehce upravené úseky popsané výše a vytvořit příklad zónového souboru pro fiktivní doménu *kdesi.cz*. O takovýto záznam se stará správce (majitel) domény a má možnost jej měnit.

\$TTL 1w

```
@ IN SOA server.kdesi.cz. franta.kdesi.cz. (200605140 1h 5m 1w 1d )
  IN NS server
  IN NS ns.jinde.cz.
  IN MX 10 server
  IN MX 20 mail.jinde.cz.
```

```
cosi IN A 1.2.3.4
    IN AAAA 2001:718:1c01:1:02e0:7dff:fe96:daa8
server IN A 1.2.3.1
www IN CNAME server
```

V příkladu je uvedena implicitní životnost jeden týden, dále SOA záznam popsaný výše, určující, že se záznam týká domény *kdesi.cz*, jejíhož správce je možné kontaktovat na emailu *franta@kdesi.cz*. Následují odkazy na dva DNS servery, které o doméně poskytují autoritativní informace – jeden místní a druhý externí. Dále MX záznamy, určující kam se bude doručovat elektronická pošta pro tuto doménu.

Další dva řádky obsahují A a AAAA záznamy určující adresy počítače *cosi.kdesi.cz*. Za nimi je definována IPv4 adresa pro *server.kdesi.cz* a konečně je mu přidělena přezdívka *www.kdesi.cz*.

## DHCP

**Dynamic Host Configuration Protocol** je v název protokolu z rodiny TCP/IP nebo označení odpovídajícího DHCP serveru či klienta. Používá se pro automatickou konfiguraci počítačů připojených do počítačové sítě. DHCP server přiděluje počítačům pomocí DHCP protokolu zejména IP adresu, masku sítě, implicitní bránu a adresu DNS serveru. Platnost přidělených údajů je omezená, proto je na počítači spuštěn DHCP klient, který jejich platnost prodlužuje.

### Charakteristika

DHCP protokol umožňuje prostřednictvím DHCP serveru nastavovat stanicím v počítačové síti sadu parametrů nutných pro komunikaci pomocí IP protokolu (tj. využívat rodinu protokolů TCP/IP). Umožňuje předávat i doplňující a uživatelsky definované parametry. Významným způsobem tak zjednodušuje a centralizuje správu počítačové sítě (například při přidávání nových stanic, hromadné změně parametrů nebo pro skrytí technických detailů před uživateli). DHCP servery mohou být sdruženy do skupin, aby bylo přidělování adres odolné vůči výpadkům. Pokud klient některým parametrům nerozumí, ignoruje je.

Typicky se pomocí DHCP nastavují tyto parametry:

- IP adresa
- maska sítě
- implicitní brána (anglicky *default gateway*)
- DNS server (seznam jedné nebo více IP adres DNS serverů)
- a další údaje, např. servery pro NTP, WINS, ...

### Zpětná kompatibilita

Protokol DHCP nebyl se svým předchůdcem BOOTP zpětně kompatibilní, což je pro internetové protokoly a vydávání RFC velmi nezvyklé. Protokol DHCP přinesl pouze možnost „pronájmu IP adresy“. Vzhledem k modularitě BOOTP protokolu ale bylo možné tuto vlastnost implementovat i do tohoto předchůdce. Zpětně nekompatibilní protokol prosadila firma Microsoft, která pro systémy Windows 95 a novější implementovala jako standardní součást pouze podporu protokolu DHCP. Pro správce tak bylo tehdy nutné společně s novou verzí stolního systému Windows nakoupit a provozovat též serverovou edici Windows NT, protože podpora DHCP byla do stávajících BOOTP serverů (typicky provozovaných na unixových systémech) implementována až se zpožděním.

### Historie

Standard DHCP vznikl v říjnu 1993 v RFC 1531 jako nástupce protokolu BOOTP, se kterým není zpětně kompatibilní. Aktualizace z roku 1997 je aktuální definicí DHCP. Protokol přinesl možnost dynamického přidělování adres (tj. na žádost klienta server přidělí IP adresu, která nemusí být pokaždé stejná), které si stanice zapůjčuje na určitou dobu a v případě potřeby zapůjčení prodlužuje. Poslední navržený standard pro DHCP v sítích IPv6 (DHCPv6) je obsažen v RFC 3315.

V současnosti je DHCP hlavním protokolem pro automatické přidělování IP adres stanicím. Protokol BOOTP se již téměř nepoužívá (kromě míst, kde je hardwarově implementován – např. do BootROM v síťových kartách).

### Princip činnosti

Klienti žádají server o IP adresu, ten u každého klienta eviduje půjčenou IP adresu a čas, do kdy ji klient smí používat (*doba zapůjčení*, anglicky *lease time*). Poté co vyprší, smí server adresu přidělovat jiným klientům.

Klient komunikuje na **UDP** portu **68**, server naslouchá na **UDP** portu **67**.

Po připojení do sítě klient vyšle **broadcastem DHCP DISCOVER** paket. Na ten odpoví DHCP server paketem **DHCP OFFER** s nabídkou IP adresy. Klient si z (teoreticky několika) nabídek vybere jednu IP adresu a o tu požádá paketem **DHCP REQUEST**. Server mu ji vzápětí potvrdí odpovědí **DHCP ACK**. Jakmile klient obdrží

DHCPACK, může už IP adresu a zbylá nastavení používat. Klient musí před uplynutím *doby zapůjčení* z DHCPACK obnovit svou IP adresu. Pokud lhůta uplyne aniž by dostal nové potvrzení, klient musí IP adresu přestat používat.

Protokol definuje roli i tzv. **DHCP relay agenta**. Používá se v situaci, kdy **existují dvě nebo více sítí oddělené směrovačem a jen jedna síť obsahuje DHCP server**. V takovém případě správce na směrovači zapne relay agenta a nastaví jej tak, aby všesměrové (*broadcast*) DHCP dotazy ze sítí bez DHCP serveru přeposílal DHCP serveru. Agent k přeposílanému dotazu přidá číslo sítě a masku sítě, na které klienta zaslechl, aby DHCP server poznal, ze kterého adresního rozsahu má klientovi adresu přiřadit.

### **Možnosti přidělení IP adresy**

IP adresa může být stanici přidělena několika způsoby:

**Ruční nastavení** - V tomto případě správce sítě nevyužívá DHCP serveru a konfiguraci jednotlivých stanic zapisuje jednotlivě přímo do konfigurace jednotlivých stanic.

**Statická alokace** - DHCP server obsahuje seznam MAC adres a k nim příslušným IP adres. Pokud je žádající stanice v seznamu, dostane vždy přidělenou stejnou pevně definovanou IP adresu.

**Dynamická alokace** - Správce sítě na DHCP serveru vymezí rozsah adres, které budou přidělovány stanicím, které nejsou registrovány. Časové omezení pronájmu IP adresy dovoluje DHCP serveru již nepoužívané adresy přidělovat jiným stanicím. Registrace dříve pronajatých IP adres umožňuje DHCP serveru při příštím pronájmu přidělit stejnou IP adresu.

V IPv6 sítích je automatickému nastavení stanice věnována vyšší pozornost, aby byla konfigurace počítačové sítě ještě jednodušší.

## **RDP**

**Remote Desktop Protocol** je síťový protokol, který umožňuje uživateli ovládat vzdálený počítač prostřednictvím připojení k jeho desktopovému prostředí. Připojení pracuje na principu klient-server, kdy uživatel na svém počítači využívá jednoduchého klienta pro zobrazení grafického uživatelského prostředí, které je spuštěno na vzdáleném počítači.

### **Historie.**

Protokol RDP byl poprvé představen ve Windows NT 4.0 Terminal Server Edition. Klienti pro připojení pomocí RDP protokolu existují pro většinu verzí Windows, Mac OS X a další operační systémy. **Server implicitně naslouchá na TCP portu 3389**. Firma Microsoft nabízí klientský software Remote Desktop Connection (RDC) nebo Terminal Services Client (TSC). V operačním systému Mac OS X se klient nazývá Remote Desktop.

### **Podporované vlastnosti**

- barevná hloubka 8, 15, 16, 24 a 32 bitů
- 128bitové šifrování algoritmem RC4 (starší klienti mohou použít slabší kódování a verze nižší než 6.0 jsou napadnutelné útokem Man in the middle)
- podpora Transport Layer Security
- Audio Redirection – přesměrování výstupu zvuku umožňuje uživateli spustit program na vzdálené ploše s výstupem zvuku na svém lokálním počítači
- File System Redirection – umožňuje použít lokální soubor umístěný na klientovi na vzdáleném počítači
- Printer redirection – umožňuje použít lokální tiskárnu pro tiskový výstup programů spuštěných na vzdáleném počítači
- Port Redirection – umožňuje použít lokální sériový a paralelní port pro programy spuštěné na vzdáleném počítači
- schránka pro kopírování textu může být sdílena mezi vzdáleným počítačem a lokálním počítačem

Ve verzi RDP 6.0 bylo v roce 2006 přidáno několik rozšíření:

- Remote programs – vzdálené programy jsou asociovány se soubory umístěnými na lokálním počítači
- Seamless Windows – vzdálené aplikace mohou běžet na počítači klienta v okně, jako by byly spuštěny lokálně
- Terminal Server Gateway – připojení prostřednictvím portu 443 (HTTPS) a IIS
- podpora Windows Aero a technologie ClearType
- podpora WPF efektů pro aplikace .NET 3.0
- přesměrování zařízení bylo přepsáno pro zajištění větší flexibility a zpřístupnění více zařízení
- všechny terminálové služby jsou plně konfigurovatelné a skriptovatelné pomocí [WMI](#)
- vylepšená adaptace RDP klientů pro aktuální šíři přenosového pásma
- šifrování pomocí Transport Layer Security (TLS) 1.0 jako implicitní pro server i klienta
- podpora více monitorů – relace může být rozdělena mezi dva monitory

### Implementace v Microsoft Windows

**Serverová část RDP** (tzv. *Terminal Services*) je součástí serverové edice Windows NT od verze NT 4.0 Terminal Edition do současných verzí (Windows Server 2003, Windows Server 2008). V omezené podobě, která neumožňuje současnou práci více uživatelů zároveň, je k dispozici i v některých verzích desktopové řady Windows NT: Windows XP (verze Professional), Windows Home Server, Media Center, Tablet PC, Windows Fundamentals for Legacy PCs, Windows Vista (ve verzích Ultimate, Enterprise a Business) a též Windows 7 (ve verzích Professional, Enterprise a Ultimate).

**Klient pro RDP** (Remote Desktop Connection, RDC a Terminal Service client) je dostupný ve všech verzích Windows XP a Windows Vista (odpovídající verzi systému nebo service packu). Starší verze klienta jsou volně dostupné pro Windows XP (pre-SP3), Windows 2000, Windows 9x, Windows NT 4.0 a Mac OS X. Poslední verze klienta 6.1 je dostupná ve Windows Vista Service Pack 1, Windows XP Service Pack 3 a jako KB952155 pro Windows XP SP2.

Ruční spuštění klienta ve Windows XP je možné pomocí příkazu `mstsc.exe`.

### Implementace v Mac OS X

Aqua Connect se stala první společností, která získala licenci a implementovala RDP pro operační systém Mac OS X, ke kterému je tak možné se připojit pomocí RDP.

### Implementace v unixových systémech

Serverová část implementace RDP pro unixové systémy se jmenuje **xrdp** a je k dispozici jako open source software. Implementace RDP klienta se jmenuje **rdesktop** a je též k dispozici jako open source software. Pro grafická prostředí lze použít jako frontend program **tsclient**.

## Bezdisková stanice (Thin Client)

**Bezdisková stanice** neboli „**tenký klient**“ (v angličtině **Thin Client**) je velice jednoduchý počítač nebo terminál sloužící k připojení se k pracovnímu počítači serveru na kterém je vykonávána celá práce uživatele. Původně se jmenoval „grafický terminál“ (v angličtině Graphical terminals). Narozdíl od „**tlustého klienta**“ (v angličtině **Fat Client**), který používá server pouze ke komunikaci a ukládání dat.

Na mnoha bezdiskových stanicích je spouštěn pouze webový prohlížeč nebo software vzdálené plochy, což znamená že ke všem významným zpracování dat dochází na serveru. Nicméně na trhu se objevují bezdiskové stanice na kterých je možné spouštět operační systémy jako je Debian Linux, čímž se zařazují mezi tzv. „hybridní klienty“. Někdy se také bezdiskové stanice nazývají „přístupové terminály“.

V důsledku toho pojem „bezdisková stanice“, zahrnuje všechny zařízení zamýšlené nebo používané, jak jsou definovány bezdiskové stanice – i když jejich schopnosti jsou mnohem větší.

### Úvod

Při navrhování počítačového systému musíme řešit otázky zpracování a ukládání dat, softwaru a uživatelského rozhraní. Se spolehlivou vysokorychlostní sítí můžeme změnit umístění těchto funkcí. Gigabitová síť je rychlejší než sběrnice PCI a mnoho pevných disků, takže každá funkce může být umístěna jinde. Výsledná volba závisí na nákladech na provoz, spolehlivost, výkon a použitelnost systému. Bezdiskové stanice jsou úzce spojeny s uživatelským rozhraním.

Na bezdiskových stanicích je nainstalován pouze software uživatelského rozhraní, některé často používané aplikace a síťový operační systém. Protože nejsou bezdiskové stanice tak zatíženy, mají menší velikost, energetickou náročnost i cenu nákupu a provozu. Všechny aplikace a služby jsou spuštěny na serveru nebo clusteru serverů a zároveň slouží i pro ukládání dat. Vytížením několika serverů a odlehčením mnoha klientů dosáhneme zjednodušení správy systému, snížení nákladů a lépe využijeme výhod sítě: centralizovaného ukládání dat/zálohy dat a snadnější zabezpečení. Vzhledem k relativní pasivitě systému a jeho snadné údržbě se systém snadněji instaluje a obsluhuje. Díky stoupajícím nákladům na hardware, na techniky, na nákup energie a na likvidaci odpadů, se bezdiskové stanice stávají výhodnější. Z pohledu uživatele se téměř nic nemění při používání myši, klávesnice a monitoru oproti tlustým klientům. Jeden PC obvykle může napájet pět a více bezdiskových stanic. Mnohem silnější PC nebo server podporuje až 100 klientů najednou, a vysoce výkonné servery podporují až 700 klientů. Do bezdiskových stanic investují často školy a podniky kteří chtějí maximalizovat počet pracovních míst, které si mohou ze svého rozpočtu dovolit.

### Historie

Bezdiskové stanice byly dříve nazývány „grafické terminály“, protože se vyvinuly z „textových terminálů“, které byly před nimi (textové terminály jsou v podstatě bezdiskové stanice, ale neříká se jim tak, protože pocházejí z jiné výpočetní doby).

### Příklady použití bezdiskových a diskových stanic

Bezdiskové stanice se mohou použít například ve školách, kde je potřeba aby velké množství studentů ve třídě pracovalo na stejném softwaru stejným způsobem. Některé společnosti nyní prodávají laptopy, které mohou přistupovat k vnitřním zdrojům pomocí virtuální privátní sítě, takže spojení klienta a serveru prochází zakódovaným tunelem. To umožňuje mobilním pracovníkům připojovat se k citlivým datům s nižším rizikem ohrožení nebo odcizení dat protože nemá žádné vlastní úložné zařízení.

### Průmyslové použití bezdiskových stanic

Od roku 2006 vzrůstá zájem o použití bezdiskových stanic v rizikových prostředích jako je průzkum ropy a zemního plynu, ve vojenství a v průmyslových oblastech Zóny 1, kde mohou být průmyslové počítače moc drahé. Bezdiskové stanice se snadněji zabezpečují proti nepříznivému prostředí a někdy mohou vydržet větší teplotní rozdíly a vibrace, díky zjednodušeným komponentám a nepřítomností pohyblivých částí jako jsou pevné disky a větráky. Bezdiskové stanice obvykle používají vestavěné operační systémy jako Linux, Windows



CE.net nebo Windows XP Embedded Ethernet. Průmysloví klienti typicky upřednostňují operační systémy které je snadné propojit s vestavěným firmware. Pro komunikaci bezdiskových stanic a monitorovaných či ovládaných zařízení je preferován Ethernet. Některé bezdiskové stanice jsou úzce spjaty se specializovaným ovládacím softwarem, který vylepšuje možnosti síťových operačních systémů. Tato vylepšení zahrnují zálohování serveru, redundantní Ethernet, více relací u jednoho klienta a automatické nastavení klienta.

### **Výhody bezdiskových stanic**

- **Nižší náklady na správu.** Bezdiskové stanice jsou téměř úplně spravovány serverem. Hardware je méně náchylný na poškození a klient je jednodušší (často bez trvalého úložiště dat), což ho chrání před malware.
- **Jednodušší zabezpečení.** bezdiskové stanice mohou být navrženy tak, že na klientovi nebudou uchovány data žádné aplikace (jsou je zobrazeny), čímž se centralizuje ochrana proti malware a snižuje se šance na fyzické odcizení dat.
- **Vylepšená bezpečnost dat.** Při vážném poškození bezdiskové stanice nemůže dojít ke ztrátě dat, protože ty jsou umístěny na serveru.
- **Nižší cena hardwaru.** Neobsahují pevný disk ani výkonný procesor a obsahují méně pohyblivých částí. Prodlužuje se doba po kterou nemusíme vylepšovat hardware, namísto toho vylepšujeme pouze síť a server. Mnoho diskových stanic se vyměňuje po třech letech aby se předešlo poškození hardwaru a zastarání softwaru, zatímco bezdiskové stanice mohou spolehlivě pracovat i deset let. Celkové nároky na hardware jsou obvykle mnohem nižší v porovnání se systémy diskových stanic, hlavně díky lepšímu využití hardwaru. Pokud více klientů používá jednu aplikaci, stačí aby byla jednou nahrána do RAM v serveru (pokud to aplikace podporuje). U diskových stanic musí mít každá svou vlastní kopii programu v paměti.
- **Menší spotřeba energie.** Typická bezdisková stanice má nižší spotřebu energie, což nejen že snižuje náklady na energii, ale také v některých případech není nutné vylepšovat chladič systém nebo nemusí být chlazení vůbec potřeba. Na druhou stranu jsou kladeny vyšší nároky na servery a síť.
- **Jednodušší oprava hardwaru.** Pokud se bezdisková stanice porouchá, lze ji jednoduše nahradit jinou a poškozenou opravit. Uživatel může pokračovat v práci protože jeho data nejsou uložena v klientovi.
- **Použitelnost v nepříznivém prostředí.** Většina bezdiskových stanic nemá žádné pohyblivé části, takže mohou být použity v prašném prostředí, kde by se za normálních okolností mohl zanést větráček a tím by došlo k přehřátí a spálení PC.
- **Nižší šířka pásma sítě.** Po síti se přenáší pouze pohyb myši, stisk kláves a aktualizace zobrazovaných informací, přes protokoly ICA nebo NX.
- **Efektivnější využití výpočetních zdrojů.**
- **Jednodušší vylepšování hardwaru.** Pokud je nutné zvednout výkon, stačí vylepšit server nebo pořídit nový, který může fungovat spolu se starým, zatímco u diskových stanic je nutné starou jednotku nahradit novou, což může zpomalit uživatele a navíc je nutné odstranit starou jednotku.
- **Nižší hluk.** Dříve zmíněné odstranění větráků snižuje množství hluku a vytváří příjemnější pracovní prostředí.

### **Výhody diskových stanic**

- **Nižší nároky na server.** Většina výpočtů se provádí na diskové stanici.
- **Vyšší multimediální výkon.** Jsou výhodnější u multimediálních aplikací a her než bezdiskové stanice, kde by tyto aplikace vyžadovali příliš velkou šířku pásma sítě.

- **Větší flexibilita.** Software pro některé operační systémy (např. Microsoft Windows) jsou vytvořené pro osobní počítače s jejich lokálními zdroji. Spuštění těchto aplikací na bezdiskových stanicích může být náročné nebo i nemožné.
- **Lepší podpora periférií.** Bezdiskové stanice jsou obvykle velmi malé, bez možnosti interního rozšíření a omezeného externího rozšíření.
- **Použitelnost i s horším síťovým připojením.** Bezdiskové stanice jsou při použití pomalého síťového připojení velmi pomalé. A bez síťového připojení nepracují vůbec.

## Aplikační vrstva modelu TCP/IP.

Vrstva aplikací - to jsou programy (procesy), které využívají přenosu dat po síti ke konkrétním službám pro uživatele. Příklady: Telnet, FTP, HTTP, DHCP, DNS.

Aplikační protokoly používají vždy jednu ze dvou základních služeb transportní vrstvy: TCP nebo UDP, případně obě dvě (např. DNS). Pro rozlišení aplikačních protokolů se používají tzv. **porty**, což jsou domluvená číselná označení aplikací. **Každé síťové spojení aplikace je jednoznačně určeno číslem portu a transportním protokolem (a samozřejmě adresou počítače).**

### Porty - body SAP

Jak již víme, na úrovni síťové vrstvy se používají síťové adresy (v TCP/IP tzv. IP adresy), které identifikují jednotlivé uzly jako celky. Přesněji: identifikují jedno síťové rozhraní, s tím že některé uzly (například směrovače) jich mohou mít více. Rozhodně ale tyto síťové adresy nerozlišují jednotlivé příjemce a odesílatele v rámci daného uzlu.

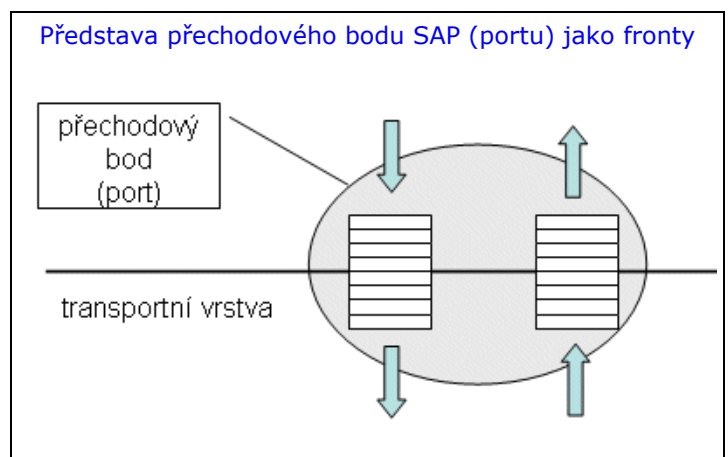
Rozlišování jednotlivých příjemců a odesílatelů má na starosti až transportní vrstva. K tomu také mu-

sí používat nějaké konkrétní adresy (transportní adresy), které by měly mít relativní charakter (měly by se vztahovat jen k danému uzlu, resp. rozlišovat jen v rámci tohoto uzlu). Jak by ale tyto transportní adresy měly vypadat?

Problém je v tom, že na různých systémových platformách (například v prostředí MS Windows, Unixu, Linuxu, MAC OS a dalších) mohou být odesílatelé a příjemci obecně různého typu - mohou to být různé procesy, eventuálně systémové úlohy, vlákna, nebo třeba i rezidentní programy (ještě ve starém MS DOSu). Jejich identifikace v rámci příslušné platformy se může i diametrálně lišit. Někdy mohou být identifikováni číslky, jindy znakovými identifikátory apod. Hlavně ale: příslušné entity (procesy, úlohy atd.) vznikají a zase zanikají dynamicky. Takže není dopředu známo, kdo (jaký proces, úloha atd.) bude kde existovat, a měl by přijímat data, určená například WWW serveru (nebo poštovnímu serveru apod.).

Všimněte si dobře jedné věci: na straně příjemce nejde ani tak o to, kdo je ten kdo nějaká data přijímá, jako spíše o to, že "ten někdo" poskytuje určitou službu - a tudíž přijímá data, určená poskytovateli příslušné služby. Tomu, kdo je v roli klienta (například WWW klienta, a vznáší požadavek na nějakou WWW stránku), může být celkem jedno, jaký konkrétní proces na druhé straně plní roli WWW serveru. Důležité pro něj je, aby někdo takový existoval a fungoval, a aby klient věděl, kam mu poslat svůj požadavek. Takže by se vlastně hodila spíše nějaká adresace ve stylu "příjemcem je ten, kdo poskytuje službu AB", místo adresace ve smyslu "příjemcem je proces s identifikátorem XY". A tak to také v praxi (ve světě RM ISO/OSI i TCP/IP) skutečně chodí.

Příslušné adresování, používané na úrovni transportní vrstvy, však není založeno přímo na představě poskytovatelů služeb. To by nemuselo být až tak šikovné, a třeba by to bylo někdy i omezující. Místo toho je použit



jiný koncept, založený na představě "přechodových bodů" mezi transportní vrstvou a vrstvou bezprostředně vyšší (což je ISO/OSI vrstva relační), a u TCP/IP vrstva aplikační. Tyto přechodové body jsou určitou abstrakcí, kterou je nejlépe si připodobnit k vyrovnávací paměti (bufferu) s režimem fronty: z jedné strany se do přechodového bodu určitá data vkládají (zapisují), a z druhé strany se z ní ve stejném pořadí vyjímají (čtou). Představu ilustruje obrázek.

Přechodové body jsou obecně obousměrné (lze si je představit jako dvě samostatné fronty s opačným "směrem"). V prostředí RM ISO/OSI se jim říká "body přístupu ke službám" (SAP, Service Access Points), ale známější asi jsou pod svým označením z prostředí TCP/IP, kde se o nich mluví jako o portech.

### Porty a procesy

Výhodou portů (přechodových bodů SAP) je tedy to, že jsou abstrakcí - a jako takové mohou být všude (na všech platformách) stejné. To má zásadní výhodu v tom, že s jejich existencí je pak možné počítat "dopředu" (apriori), stejně jako s tím, že mají stejné vlastnosti. Konkrétní způsob jejich realizace na každém jednotlivém uzlu sítě naopak může zůstat skryt.

Jednotné (všude stejné) může být i označení portů, resp. bodů SAP, a tím i jejich adresování (na úrovni transportní vrstvy). V

prostředí TCP/IP jsou jednotlivé porty adre-

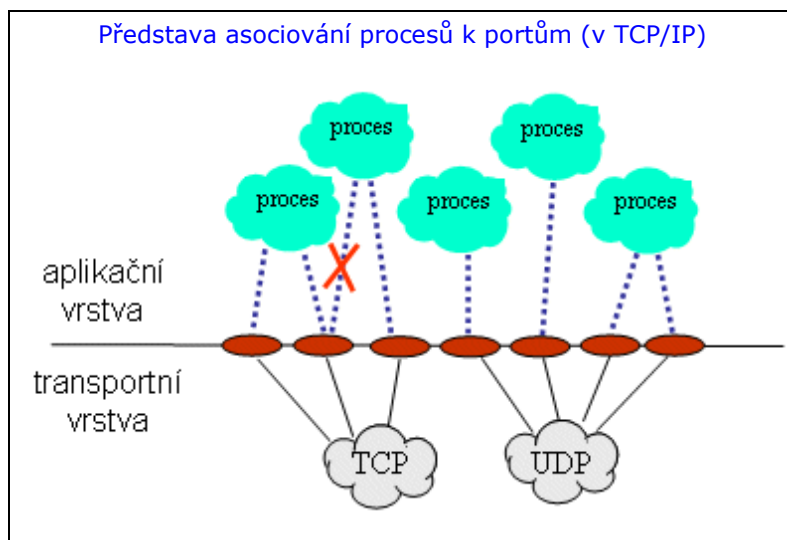
sovány skrze pořadová čísla. Přesněji: pomocí celých nezáporných čísel (0, 1, 2, 3, 4 atd.). V praxi se pak hovoří o číslech portů. Třeba o portu číslo 80, o portu č. 25 apod. Takže transportními adresami (relativními adresami na úrovni transportní vrstvy) jsou v TCP/IP. O právě tato čísla portů.

Důležité ale je uvědomit si, že port (bod SAP) ještě není to samé, jako proces (úloha, vlákno atd.), které skutečně přijímá a zpracovává nějaká data, nebo je naopak generuje. Port je skutečně jen jakýsi "průchozí bod" (s režimem fronty), za kterým je teprve "schována" příslušný proces (úloha atd.). Ovšem když porty existují "dopředu" (apriori), zatímco aplikační entity (procesy, úlohy) vznikají dynamicky, podle momentálních potřeb, musí i jejich vzájemná vazba být dynamická. V praxi to vypadá tak, že když je vytvořen nějaký proces, a ten má nějak komunikovat v prostředí sítě, musí se nejprve "sdružit" (tzv. asociovat) s některým portem. Teprve pak může odesílat svá data (skrze tento port), a také je (skrze tento port) přijímat. Protistrana, která s takovýmto procesem komunikuje, musí znát číslo příslušného portu, a svá data také adresuje tomuto portu (přesněji: na daný cílový uzel, a v rámci tohoto cílového uzlu příslušnému portu).

Obecně přitom platí, že jeden proces (úloha atd.) může být asociován (sdružen) s více porty. Naopak to ale neplatí: s jedním portem může být sdružen (asociován) nejvýše jeden proces. Důvod je jednoduchý: copak by asi takový port dělal, kdyby přijal nějaká příchozí data? Jak by vybíral mezi dvěma (či dokonce více) procesy, kterým by tato data měl předat?

### Servery a jejich klienti

Ukažme si nyní na konkrétním příkladu, jak celá představa o portech a procesech zapadá do běžné praxe, kdy spolu v prostředí sítě spolu komunikují aplikace a procesy v roli serverů a jejich klientů. Třeba když si uživatel na svém osobním počítači pustí WWW browser, a tomu zadá, že chce navštívit nějakou konkrétní WWW stránku, např. na adrese [www.earchiv.cz](http://www.earchiv.cz) (kde má svůj osobní archiv autor tohoto seriálu). Webový browser si nejprve nechá přeložit symbolické doménové jméno ([www.earchiv.cz](http://www.earchiv.cz)) na odpovídající IP adresu, a na ni pak pošle požadavek na stažení konkrétní stránky. Na cílovém počítači je ale třeba předat příslušný požadavek tomu procesu, který zde plní roli WWW serveru.



Klient (uživatelův browser) přitom dopředu ví, že takovýto proces je sdružen (asociován) s portem číslo 80, a tak jej adresuje právě tomuto portu. Proces, který na adrese [www.earchiv.cz](http://www.earchiv.cz) funguje jako WWW server, tak požadavek dostane, a v odpovědi na něj zašle klientovi požadovanou WWW stránku (index.php3). Přitom také musí uvést nějaké číslo portu, na které má být odpověď klientovi doručena. Server tedy standardně posílá svou odpověď na stejný port (na straně klienta), ze které mu přišel původní požadavek.

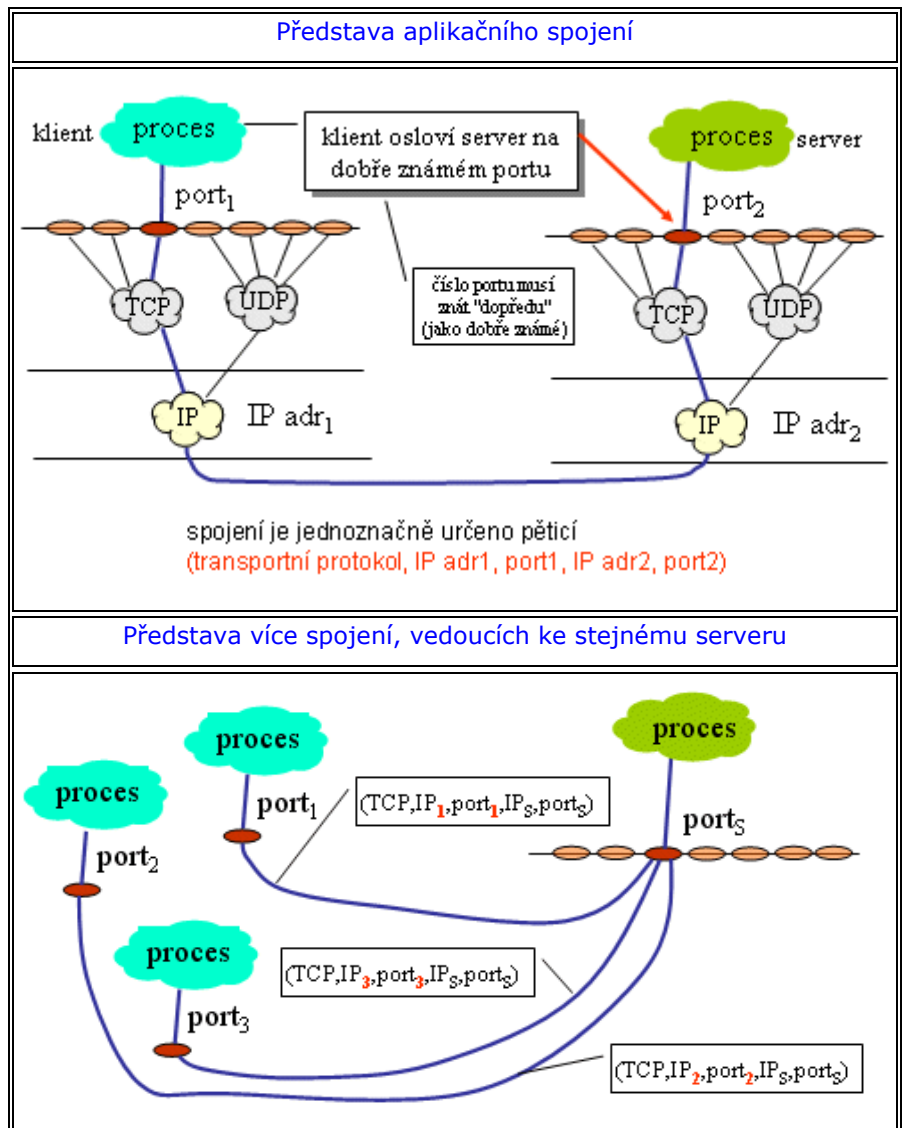
Celou situaci s komunikací klienta a serveru názorně ukazuje obrázek, kterým si můžeme znovu připomenout jiný úkol transportní vrstvy, popisovaným v úvodu: měnit nespolehlivý a nespojovaný způsob komunikace (na úrovni síťové vrstvy, pomocí protokolu IP) na spojovaný a spolehlivý (na úrovni transportní vrstvy, pomocí protokolu

TCP), nebo jej ponechávat beze změny, jako nespolehlivý a nespojovaný (pomocí protokolu UDP) - podle toho, co si vybere příslušná aplikace. Přitom služba World Wide Web (WWW) je příkladem aplikace, která dává přednost spojovanému a spolehlivému přenosu, a využívá tedy služeb transportního protokolu TCP.

Jen pro ilustraci si naznačme ještě to, jak může jeden WWW server současně poskytovat své služby, skrze jeden a ten samý port (číslo 80), více klientům. Když mu všichni posílají své požadavky na stejnou adresu (IP adresa, port č. 80), jak pozná od koho, co pochází a kam má zaslat požadovanou WWW stránku? Odpověď je principiálně jednoduchá a stejná **pro všechny druhy serverů: u každého požadavku si pamatují, ze kterého spojení přišel. A toto spojení je obecně určeno pěticí údajů:**

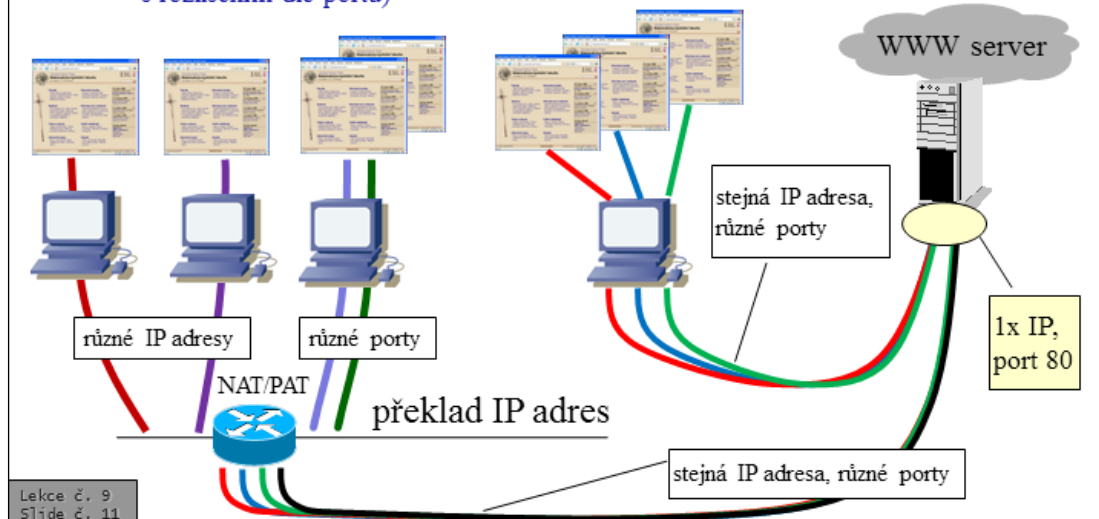
- transportním protokolem (v případě WWW jde o TCP)
- IP adresou a portem na straně klienta
- IP adresou a portem na straně serveru

Takže když server dostane dva požadavky od klientů na různých počítačích, liší se příslušná spojení už IP adresou klienta. A pokud by snad šlo o požadavky od dvou různých klientů, běžících na stejném počítači (například když má uživatel puštěny dva browsery, případně jen dvě okna či záložky ve stejném browseru), rozliší je alespoň podle různého čísla portu, použitého na straně klienta. Vše ilustruje i následující obrázek



## představa (aplikačního) spojení

- jedna entita (proces, ... ) může komunikovat s více entitami na stejném uzlu, i na různých uzlech za jedním NAT/PAT uzlem
  - více entit na stejném uzlu: když si uživatel spustí více instancí browseru (například)
  - více entit za uzlem NAT/PAT (který zajišťuje překlad více různých adres do jedné, s rozlišením dle portů)



Lekce č. 9  
Slide č. 11

V případě, že komunikace mezi klientem a serverem probíhá nespojovaně (prostřednictvím transportního protokolu UDP), je identifikace odesílatele a příjemce obdobná. Server dokáže rozlišit požadavky různých klientů podle IP adresy a čísla portu (na straně odesílatele), i když zde žádné spojení vůbec nevzniká.

### Konvence o tzv. dobře známých portech

Zastavme se na závěr u jedné důležité drobnosti: jak může webový klient (browser) dopředu vědět, že své požadavky má posílat "svému" serveru na port číslo 80? Jak může vědět poštovní klient, že má odesílanou poštu posílat "svému" serveru na port číslo 25? Jak je tomu pro ostatní služby a aplikace?

Odpověď je taková, že musí existovat určitá konvence, ve výše naznačeném smyslu, a ta musí být dopředu známá všem, kteří ji potřebují znát. Například když WWW klient (browser) dostane od svého uživatele pokyn k načtení nějaké konkrétní WWW stránky, automaticky předpokládá, že jej má poslat serveru na port číslo 80, a tak to také dělá.

*Samozřejmě, pokud by uživatel měl jiné přání, a explicitně by si vyžádal zaslání svého požadavku na nějaký jiný port (např. na port č. 8080, skrze URL ve tvaru <http://www.earchiv.cz:8080/index.php3>), browser by to respektoval. Ovšem aby takový požadavek mohl být korektně přijat a vyřízen, musel by i na cílovém počítači existovat WWW server, asociovaný s příslušným portem (zde: s portem číslo 8080). Při nějakém testování či ladění tomu tak může být, při běžném rutinním provozu to ale už není zvykem (a WWW server "sedí" na standardním portu č. 80).*

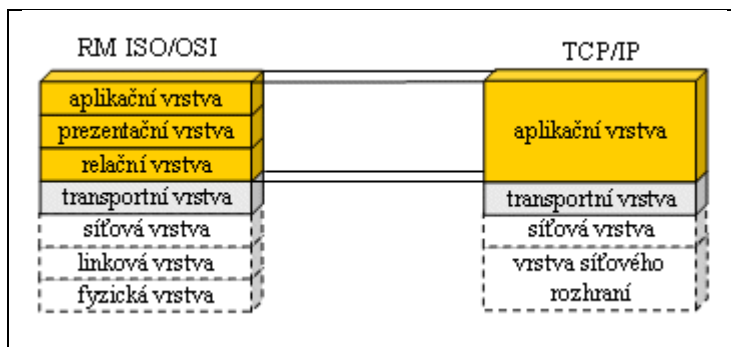
Portům, jejichž význam je dopředu stanoven, se v TCP/IP říká příznačně: dobře známé porty (anglicky: well-known ports). Jde obecně o porty v rozsahu od 0 do 1023, a příslušnou konvenci spravuje orgán jménem IANA (Internet Assigned Numbers Authority). Dříve ji publikoval pravidelně (jednou za rok, později každých 6 měsíců) ve formě dokumentu RFC. Dnes je tato konvence zveřejňována již průběžně, na WWW stránkách IANA. Malou část této konvence vidíte v tabulce.

<b>21</b>	<b>FTP</b>	<b>110</b>	<b>POP3</b>
<b>23</b>	<b>Telnet</b>	<b>119</b>	<b>NNTP</b>
<b>25</b>	<b>SMTP</b>	<b>143</b>	<b>IMAP</b>
<b>69</b>	<b>TFTP</b>	<b>161</b>	<b>SNMP</b>
<b>70</b>	<b>Gopher</b>		
<b>80</b>	<b>HTTP</b>		
<b>88</b>	<b>Kerberos</b>		



## Odlišnosti modelů v aplikačně orientovaných vrstvách.

Jedním z významných rozdílů mezi Referenčním modelem ISO/OSI a TCP/IP je i počet "nejvyšších" vrstev. Zatímco referenční model ISO/OSI má samostatnou prezentační a relační vrstvu, v TCP/IP je nenajdeme. Pokud jsou jejich funkce zapotřebí, jsou implementovány až v samotné aplikační vrstvě. Tu samozřejmě najdeme jak u RM ISO/OSI, tak i u TCP/IP. Ovšem



představa, že do aplikační vrstvy spadají celé aplikace, používané koncovými uživateli, není zcela správná.

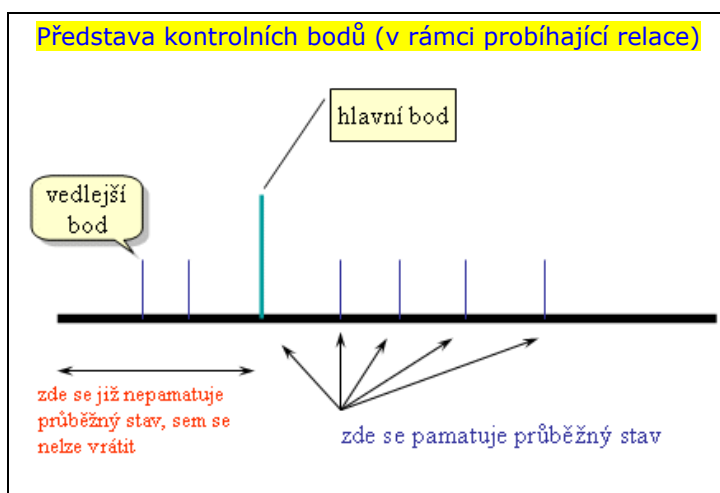
Jsou to vrstvy, nacházející se nad vrstvou transportní, kterou jsme se zabývali minule. Jak vidíme na dnešním prvním obrázku, počet těchto vrstev se v ISO/OSI a TCP/IP výrazně liší.

Odpověď je taková, že autoři obou architektur přemýšleli o tom, zda a jak často budou aplikace potřebovat určité funkce (prezentační a relační funkce, viz dále). Autoři modelu ISO/OSI dospěli k závěru, že spíše častěji - a tak považovali za správné poskytnout tyto funkce všem aplikacím, právě formou samostatné prezentační a relační vrstvy. Naopak autoři TCP/IP dospěli při svých výchozích úvahách k závěru, že využití těchto funkcí nebude až tak časté, a tak se nevyplatí vnucovat je každému (ve formě samostatných vrstev). A pokud je některá aplikace potřebovat bude, ať si je raději zajistí (implementuje) sama, podle svých vlastní představ, ale hlavně ve vlastní režii.

Právě režie je totiž faktorem, který v uvedených úvahách sehrál významnou roli. Každá vrstva už svou samotnou existencí představuje určitou režii, vynakládanou na to, aby data "prošla" skrze takovouto vrstvu. Pokud by, ve většině případů, tato vrstva vlastně s daty nic nedělala, pak by režie na průchod vrstvou byla úplně zbytečná. Proto autoři TCP/IP raději příslušné vrstvy nezařadili do svého vrstevnatého modelu (když usoudili, že funkce těchto vrstev nebudou příliš často využívány). Naopak autorům modelu ISO/OSI vyšlo, že když bude příslušné funkce využívat většina aplikací, režie na existenci samostatných vrstev bude účelně vynaložená.

### Relační vrstva a relační služby

Relační vrstva je v ISO/OSI asi nejvíce kritizovanou vrstvou, kvůli tomu, že toho má relativně nejméně na práci. Původní představa autorů zřejmě byla taková, že **v této vrstvě budou soustředěny takové funkce, které usnadňují a podporují vzájemnou interakci komunikujících stran (tzv. relace, anglicky: sessions)**. Například ve smyslu zajištění bezpečnosti takovéto interakce. Nebo ve smyslu podpory transakčního zpracování, kde je **důležité, aby se žádná transakce neprovedla "jen tak napůl", ale vždy jen úplně celá (nebo se naopak neprovedlo vůbec nic)**. Příkladem může být



nějaká finanční transakce, spočívající v odepsání určité částky z jednoho účtu, a připsání téže částky na jiný účet. Tady by skutečně nebylo korektní, aby proběhla jen jedna část transakce (odepsání částky z jednoho účtu), ale už se neprovedly ostatní části transakce (připsání na jiný účet).

Již na tomto prvním příkladu je dobře patrné hlavní dilema: mají být takovéto věci, jako je podpora transakcí, řešeny samostatně, na úrovni relační vrstvy, a tudíž pro všechny aplikace stejně? Nebo je vhodnější, aby si je aplikace řešily samy, podle svých vlastní představ? Tady je vhodné si uvědomit, že třeba bankovní aplika-



ce asi budou mít podstatně přísnější požadavky na "kvalitu" zajištění transakcí, než nějaké jednodušší databázové aplikace, kde nehrozí tak velké následky při nekorektním provedení transakce. I v tomto ohledu zřejmě autoři TCP/IP usoudili, že je lepší, když si podporu transakcí udělá každá aplikace sama, podle svého (hlavně podle svých nároků a požadavků), zatímco autoři RM ISO/OSI byli opačného názoru. Současná praxe dává celkem jednoznačně za pravdu autorům TCP/IP.

Původně uvažované úkoly relační vrstvy samozřejmě nekončily u již zmiňované bezpečnosti a transakčního zpracování. Zahrnovaly například podporu vzájemné synchronizace komunikujících stran (ovšem v jiném smyslu, než jak jsme mluvili o synchronizaci na úrovni fyzické vrstvy). Zde jde kupř. o to, že když při nějakém přenosu většího objemu dat mezi dvěma stranami dojde k výpadku spojení, musí se začít úplně od zno-va, tzn. navázat nové spojení a přenášet data od začátku. Relační vrstva však může nabízet výhodnější možnost: vrátit se k určitému "kontrolnímu bodu" a pokračovat od něj a nikoli od začátku. Takovýchto kontrolních bodů může existovat více, a v přerušené relaci pak lze pokračovat od kteréhokoli z nich. Vlastně ne nutně od kteréhokoli, protože to by bylo moc drahé. Jednotlivé kontrolní body jsou totiž jakési body zotavení, v rámci kterých je uschován celý stav právě probíhající relace (včetně dosud přenesených dat). A to představuje dost velkou režii, kvůli které se starší kontrolní body již mohou "uvolňovat" a není možné se k nim vracet. Situaci naznačuje následující obrázek, s vedlejšími a hlavním kontrolním bodem (hlavní je ten kontrolní bod, přes který se již nelze vrátit hlouběji do historie relace, ke starším vedlejším bodům).

Do relační vrstvy by nejspíše patřily i takové služby, jaké dnes v rámci TCP/IP zajišťuje protokol SIP. Ten má podporu relací dokonce ve svém názvu, protože jeho jméno je zkratkou od Session Initiation Protocol (doslova: podpora navazování relací). Je z oblasti internetové telefonie, a službám na bázi VOIP slouží k navazování telefonických hovorů. Řeší zejména takové věci, jako je vyhledání volaného (podle telefonního čísla), tak aby k němu mohlo být navázáno transportní spojení a následně veden hlasový hovor. Příznačné ovšem je, že v rámci TCP/IP jde o aplikační protokol - protože, jak již víme, TCP/IP nemá samostatnou relační vrstvu. A docela dobře se bez ní obejde.

### **Prezentační vrstva a služby**

Zatímco u relační vrstvy není úplně snadné vysvětlit, co by vlastně měla mít na starosti, v případě prezentační vrstvy je to naštěstí snazší. Stačí vyjít z následující úvahy: všechny nižší vrstvy (pod prezentační vrstvou) se snaží přenášet data tak jak "stojí a leží", aniž by se jakkoli změnila. Můžeme si to představit také tak, že přenáší skupiny bitů a snaží se, aby se při přenosu nezměnil ani jeden bit. Jenže to nemusí být vždy správné! Proč?

Třeba proto, že ona data mají určitý význam. Může to být například text, mohou to být čísla, nebo nějaké obecnější datové struktury, od jednoduchých vícerozměrných polí až po struktury provázané pointery. Ovšem na různých platformách mohou být stejné texty, stejná čísla, či stejné datové struktury reprezentovány úplně jinak. Třeba jen u textů může být rozdíl v tom, zda jsou jednotlivé znaky kódovány v kódu ASCII, v kódu EBCDIC, nebo v nějakém úplně jiném kódu (Unicode atd.). Nehledě již na různé národní znakové sady (CP xy atd.). Asi není těžké nahlédnout, že v takovém případě musí být přenášené texty vhodně překódovány (obecně: konvertovány, pomocí vhodné konverze). A právě to má na starosti vrstva prezentační: stará se o to, aby přenášená data měla stejný význam pro obě komunikující strany.

### **"Pseudovrstvy" v TCP/IP**

Jak jsme si již uvedli výše, samostatná relační a prezentační vrstva je pouze v Referenčním modelu ISO/OSI, ale v TCP/IP nikoli. Zde se předpokládá, že ta aplikace, která relační či prezentační služby potřebuje, si je zajistí sama, ve vlastní režii. O důvodech jsme se také již zmiňovali (jde hlavně o úsporu režie na samostatné vrstvy). Přesto ale i v TCP/IP existuje jakési alternativní řešení, které velmi připomíná rčení o "zlaté střední cestě" mezi dvěma extrémy (existencí samostatných vrstev a jejich úplnou absencí).

Jde o řešení, vzniklé v souvislosti s protokolem **NFS (Network File System)**, který v rámci TCP/IP slouží ke sdílení souborů. Pochází pod firmy Sun Microsystems, a bylo to vlastně úplně první řešení, které vzniklo u komerční firmy (tj. jako její vlastní, tj. "proprietární"), ale pak bylo otevřeno a mohlo se stát otevřeným internetovým standardem. Hlavně ale: i tato aplikace potřebovala zajistit určité relační a prezentační funkce - a tak si je, v duchu koncepce TCP/IP, musela zajistit sama. Ale když už tak udělala, učinila tak způsobem, který je využitelný i jinými aplikacemi, tak aby tyto již nemusely znovu "vynalézat kolo" a znovu si vyvíjet vlastní prezentační a relační funkce.

Autoři NFS totiž navrhli vše tak, že **samotný protokol NFS je "čistě aplikační"**, a má "k ruce" dva další samostatné protokoly, které pro něj zajišťují prezentační a relační funkce. Konkrétně jde o **protokoly XDR (eXternal Data Representation), a RPC (Remote Procedure Call)**.

Tyto protokoly jsou koncipovány tak, aby mohly být řešeny jako knihovní moduly, které si ostatní aplikace přilinkují, pokud je chtějí využívat - ale pokud je využívat nechťejí, nenesou režii spojenou s jejich existencí (v podobě samostatné vrstvy).

Celý efekt naznačuje následující obrázek. Vůči těm protokolům, které protokoly XDR a RPC využívají, se tyto chovají obdobně jako samostatné vrstvy. Ovšem ostatní aplikace jimi neprocházejí, a tím ani nenesou režii, která by s tím byla spojena.

### Aplikace v aplikační vrstvě?

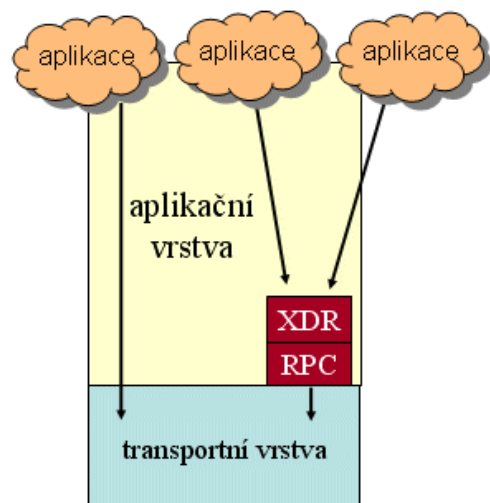
Původní představa autorů Referenčního modelu ISO/OSI možná byla taková, že aplikační vrstva bude sloužit k tomu, aby v ní byly provozovány jednotlivé aplikace. Na první pohled to vypadá velmi logicky, ale přináší to jeden zásadní problém: pokud by tomu tak bylo, pak by i všechny aplikace musely být standardizovány, a tudíž všechny podřízeny stejným pravidlům, které by definovaly nejen jejich činnost, ale i vše co s tím souvisí. Ale to by nebylo dobře.

*I aplikace určitě musí respektovat určité konvence, a musí se jim důsledně přizpůsobovat. Třeba aplikace, které se účastní práce s elektronickou poštou, musí dodržovat formátování jednotlivých zpráv, musí používat stejné koncipované adresy, stejné mechanismy přenosu zpráv atd. Ale proč by současně měly nabízet jednotné (rozuměj: úplně stejné) uživatelské rozhraní? Proč by měly svým uživatelům vnucovat vždy přesně stejný (standardizovaný) repertoár funkcí? V čem by se pak lišily? Nebo jinak, z opačného pohledu: má smysl standardizovat vzhled uživatelského rozhraní, a předepisovat autorům poštovních klientů, jak mají jejich produkty vypadat? Jaká mají používat okénka, jaké barvy, písma, repertoáry funkcí atd.? Určitě ne.*

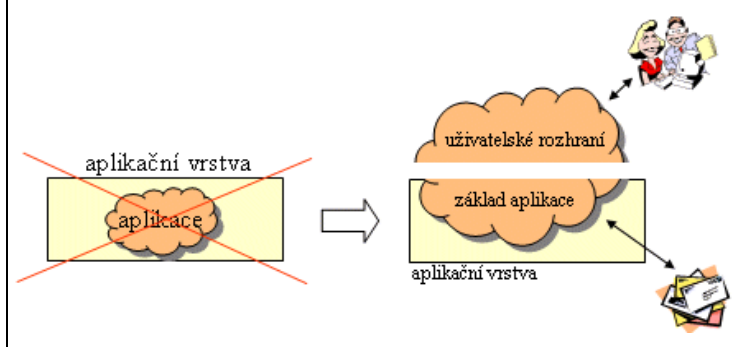
*Takže původní představa, o tom že v aplikační vrstvě budou "umístěny" celé aplikace, vzala brzy za své. Místo toho došlo k jakémusi "roztržení" aplikací na dvě části:*

1. na část, které musí být standardizována, proto aby si rozuměla s dalšími aplikacemi, resp. instancemi stejné aplikace (např. s poštovními servery atd.)
2. na část, kterou nemá smysl standardizovat (typicky jde o uživatelské rozhraní aplikace).

Představa protokolů XDR a RPC v TCP/IP



Představa vztahu aplikací a aplikační vrstvy



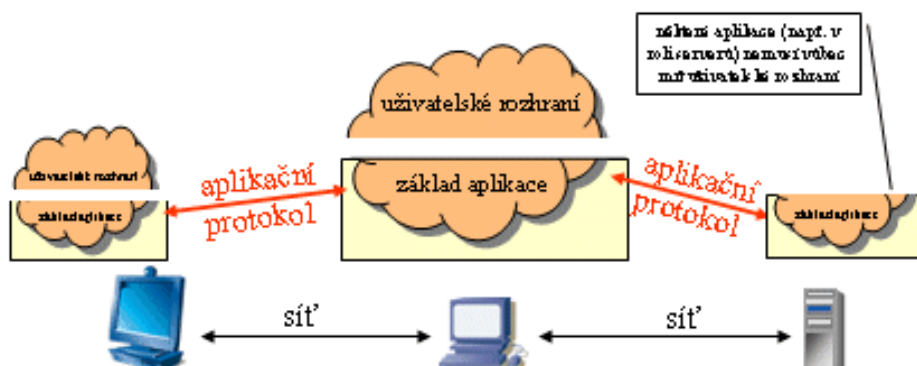
Takže původní představa, o tom že v aplikační vrstvě budou "umístěny" celé aplikace, vzala brzy za své. Místo toho došlo k jakémusi "roztržení" aplikací na dvě části:

Jak také naznačuje následující obrázek, v aplikační vrstvě nakonec zůstala jen první z obou výše uvedených částí, zatímco ta druhá již byla "vytlačena" nad aplikační vrstvu, a tím i z dosahu standardizace. Díky tomu si dnes může každý vybrat například takového poštovního klienta, jaký vyhovuje jeho potřebám - s maximálním komfortem nebo naopak jednoduchého, s grafickým uživatelským rozhraním nebo v řádkovém režimu atd.

## Aplikace a aplikační protokoly

Rozdělení aplikací na dvě části a standardizace těch částí, které zůstaly v aplikační vrstvě, ještě nestačí k tomu, aby aplikace fungovaly skutečně tak, jak mají. Například půjde-li o již zmiňovanou elektronickou poštu, musí být nějak vyřešeno předávání zpráv mezi jednotlivými uzly, které se na fungování celého "elektronického poštovního systému" podílejí. Takové předávání zpráv bude řízeno vhodným protokolem, který bude na přenos zpráv specializován.

Obecně se takovéto protokoly označují přívlastkem "aplikační", protože fungují v rámci aplikační vrstvy a slouží potřebám aplikací. Konkrétně se pak hovoří například o aplikačním protokolu pro přenos elektronické pošty (kterým je v TCP/IP) **protokol SMTP (Simple Mail Transfer Protocol)**, o aplikačním protokolu pro přenos WWW stránek (kterým je v TCP/IP) **protokol HTTP** atd.

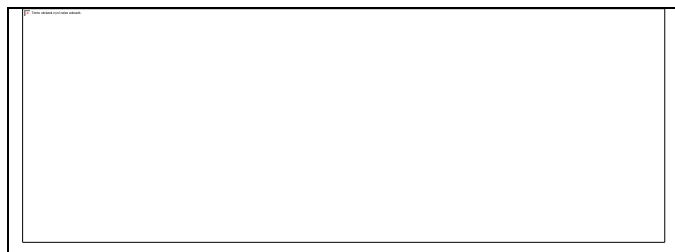


například aplikace (např. v rozhraní) nemusí vůbec mít vztah k síti

Jeho podstatou není nic jiného, než že úkoly, které mají být splněny v rámci nějaké služby, například v rámci elektronické pošty či World-Wide Webu, se rozdělí mezi dvě aplikace - jednu, která plní roli serveru, a druhou, která plní roli klienta. Proto také označení celého tohoto řešení (tzv. výpočetního modelu) termínem "klient/server".

## Architektura aplikací a výpočetní model

S aplikačními protokoly úzce souvisí také architektura samotných aplikací. Zde si pouze naznačme, že v prostředí počítačových sítí (a také celosvětového Internetu) je **dnes nejčastějším výpočetním modelem model klient/server**. Jeho podstatou není nic jiného, než že úkoly, které mají být splněny v rámci nějaké služby, například v rámci elektronické pošty či World-Wide Webu, se rozdělí mezi dvě aplikace - jednu, která plní roli



### Představa modelu klient/server v rámci služby WWW



serveru, a druhou, která plní roli klienta. Proto také označení celého tohoto řešení (tzv. výpočetního modelu) termínem "klient/server".

**Úkolem klienta** přitom je zajišťovat styk s uživatelem. Tedy zejména zobrazovat mu požadované výstupy, a přijímat od něj jeho požadavky (vstupy, obvykle zadávané z klávesnice, myši atd.). **Úkolem serveru** pak je zajišťovat vlastní zpracování, které je v rámci služby zapotřebí, a to až na výzvy klienta (resp. jeho uživatele). Asi nejnázornější je to na příkladu služby WWW: její server je tím, kdo u sebe uchovává jednotlivé WWW stránky. Sám je ale nikomu nevnučuje. Pouze pasivně čeká, až si některý WWW klient (označovaný také jako browser, či prohlížeč) o nějakou WWW stránku řekne. Pak mu ji server poskytne - a je už na

klientovi, aby tuto stránku korektně zobrazil svému uživateli. Když ten si pak vyžádá nějakou jinou WWW stránku (ať již explicitním zadáním její adresy či kliknutím na hypertextový odkaz v právě zobrazované stránce), WWW klient pošle WWW serveru nový požadavek - a vše se opakuje.

Ovšem **aby si server a klient správně rozuměli a dokázali spolu spolupracovat** tak jak je třeba, musí mezi nimi existovat **dvě konvence**:

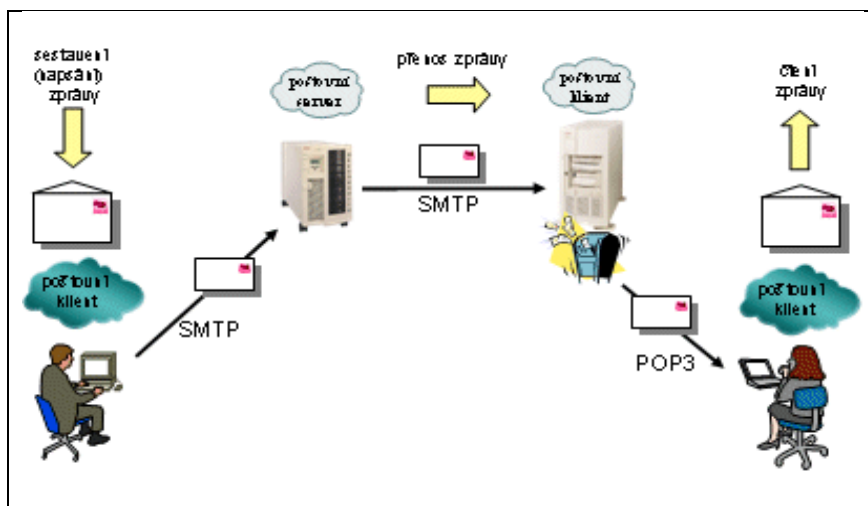
- konvence o tom, **jak budou vzájemně komunikovat**. Tuto konvenci naplňuje **aplikační protokol**, v konkrétním případě služby WWW jde o protokol HTTP (Hypertext Transfer Protocol)
- konvence o tom, **jak budou formátovány a jaký význam budou mít data**, která si klient a server vzájemně předávají. V případě WWW jde hlavně o formát WWW stránek, které server zasílá klientovi - ty jsou kódovány v jazyku HTML (HyperText Markup Language). Klient mu musí rozumět natolik, aby podle něj dokázal vytvořit grafickou podobu příslušné WWW stránky (provést tzv. rendering, neboli jakousi "vizualizaci" WWW stránky).

### Elektronická pošta jako aplikace

Ukažme si vše ještě na jednom příkladu, a to pro elektronickou poštu (v Internetu). Ta je službou, na jejíž realizaci se podílí poštovní servery (mail servery), zajišťující přenos jednotlivých zpráv, a dále poštovní klienti, kteří uživatelům zprostředkovávají čtení i psaní zpráv. Pro svou vzájemnou komunikaci přitom poštovní servery a poštovní klienti mají k dispozici více aplikačních protokolů. **Nejčastější je dnes asi stále řešení, kdy se pro odesílání zpráv (od poštovního klienta k poštovnímu serveru) a pro přenos zpráv (mezi poštovními servery) používá protokol SMTP (Simple Mail Transfer Protocol), a pro stahování zpráv (z poštovních schránek, umístěných na poštovním serveru, do poštovního klienta) se používá protokol POP3.** Proč tomu tak je, a proč zde nestačí jen jediný protokol (SMTP), či kde a jak se používají další "poštovní" aplikační protokoly jako je IMAP, si povíme později.

Teď jen sumarizující obrázek naznačující představu serverů a klientů v rámci celého systému elektronické pošty, a také jedno malé upřesnění: World Wide Web i elektronická pošta jsou službami, a jako takové jsou realizovány (implementovány) konkrétními aplikacemi, s využitím konkrétních aplikačních protokolů atd. V případě WWW existuje jen jeden způsob implementace této služby (ten, který jistě dobře znáte z Internetu, s protokolem HTTP, jazykem HTML atd.).

Ovšem u elektronické pošty je to jinak. Ta je službou, která může být (a je) implementována více různými způsoby, s využitím různých aplikačních protokolů a dalších standardů, na různých platformách atd.



Vlastně je možné hovořit o celých ucelených "systémech elektronické pošty", které se shodují v základním účelu (možnosti elektronické poštovní korespondence), ale už se mohou i významně lišit v tom, jak konkrétně to dělají. Ten systém elektronické pošty, který jsme si až dosud popisovali a který se používá v Internetu, je dnes sice dominující, ale není zdaleka jediný. Pokud je třeba jej odlišit od ostatních systémů elektronické pošty, označuje se neformálně jako "SMTP pošta" (kvůli aplikačnímu protokolu SMTP pro přenos zpráv).

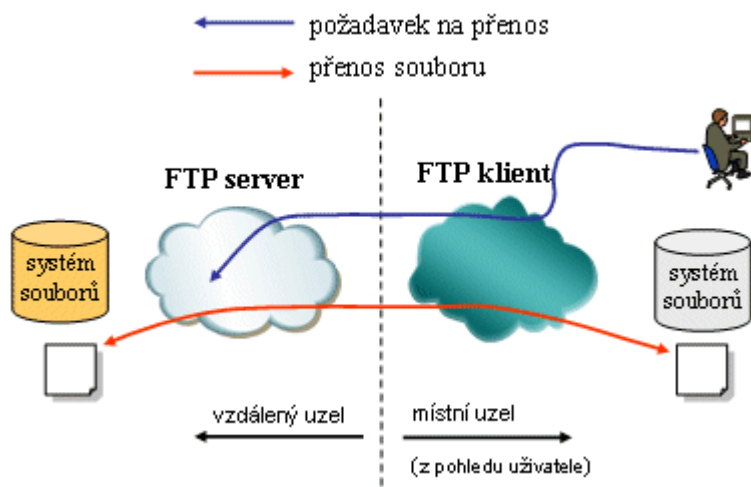
## Přenos souborů

Pro přenos souborů vznikl ve stejné době **protokol FTP** (což je zkratka od: File Transfer Protocol), umožňující přenášet celé soubory mezi jednotlivými uzly v rámci sítě. Opět ale jde o službu, zajišťovanou pomocí aplikací

fungujících v modelu klient/server. Aplikace v roli FTP serveru je "tím, kdo má soubory", zatímco klient je "tím, kdo chce soubory".

FTP server běží na nějakém uzlu (z pohledu uživatele celé služby jde o vzdálený uzel), a nabízí ke stažení všechny nebo některé soubory, nacházející se na tomto (vzdáleném) počítači. Stejně tak může nabízet nahrání (uchování, uložení) dalších souborů na uzel, na kterém běží. FTP klient pak běží na uzlu, na kterém pracuje uživatel, a skrze tohoto klienta si stahuje soubory ze (vzdáleného)

FTP serveru, nebo naopak nahrává (tzv. uploaduje) své soubory na (vzdálený) FTP server. Lze si to představit také tak, že FTP server vytváří jakési "okno", skrze které jsou "vidět" soubory, umístěné na vzdáleném počítači, a tyto jsou dostupné pro čtení a zápis. Naznačuje to i obrázek.

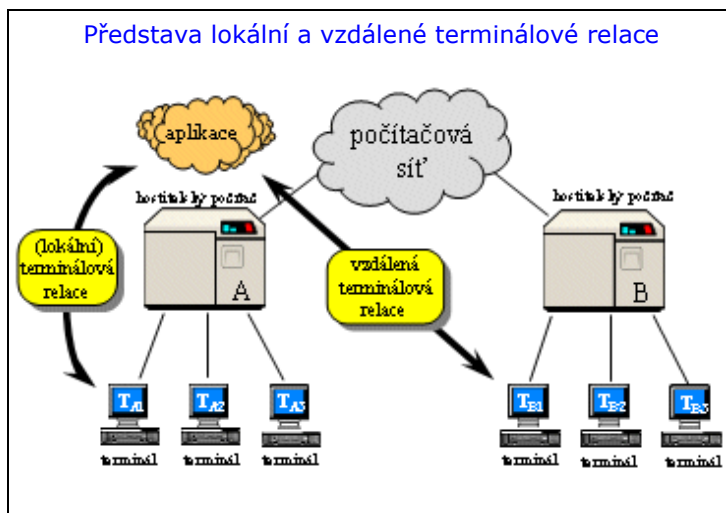


## Vzdálené přihlašování – terminálová relace

Pokud jde o vzdálené přihlašování, to je služba motivovaná zejména potřebou využívat na dálku výpočetní kapacitu a další zdroje vzdálených uzlů. Třeba k tomu, aby správce sítě mohl provést na dálku nějaké úkony (například nějaké změny nastavení) na vzdáleném počítači, a to přímo ze svého počítače, aniž by se musel fyzicky přemisťovat ke vzdálenému počítači. Také běžný uživatel si může skrze vzdálené přihlašování spustit aplikace na vzdáleném počítači, a používat je "na dálku". Je to možné díky tomu, mezi jeho počítačem a vzdáleným počítačem vzniká tzv. vzdálená terminálová relace (jako alternativa k "lokální"

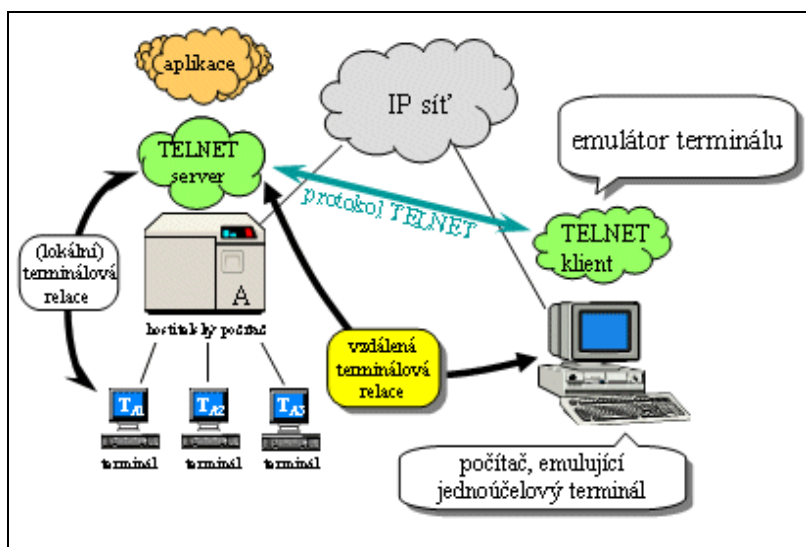
terminálové relaci). Její podstatou je přesměrování výstupů (a také vstupů) příslušné aplikace, běžící na vzdáleném uzlu. Místo toho, aby tato aplikace zobrazovala své výstupy na "místním" monitoru (resp. terminálu), jako u lokální terminálové relace, posílá je po síti na jiný uzel (kde se nachází uživatel), a teprve zde se tyto výstupy zobrazují. Obdobně se vstupy, které generuje uživatel - generuje je (mačkáním kláves na své klávesnici) tam, kde se nachází, a tyto vstupy jsou po síti zasílány až ke vzdálenému uzlu, kde běží příslušná aplikace, a jsou jí předávány.

Způsob, jakým je služba vzdáleného přihlašování realizována v prostředí TCP/IP, ukazuje následující obrázek. Opět jde o řešení na bázi modelu klient/server, s tím že komunikaci mezi klientem a serverem zajišťuje protokol Telnet. Podle něj se také hovoří o **Telnet serveru a Telnet klientovi**.





Telnet server "sedí" na vzdáleném uzlu, a má za úkol přeměňovat výstupy zdejších aplikací na jiné uzly (kde běží Telnet klient). Úkolem Telnet klienta je naopak takovéto výstupy zobrazovat uživateli na jeho displeji. Ten dříve patřil skutečnému terminálu, a proto se také hovořilo o "terminálových relacích". Dnes již uživatelé pracují téměř výlučně s běžnými univerzálními počítači, které funkce a chování jednoúčelového terminálu pouze předstírají (tzv. emulují), pomocí softwarových prostředků. Tuto roli, resp. úkol (tj. softwarovou emulaci terminálu) také plní Telnet klient. Kromě toho samozřejmě musí "sbírat" vstupy od svého uživatele (z klávesnice, případně i od myši), a tyto zase zasílat opačným směrem, k Telnet serveru. Úkolem Telnet serveru je pak "podstrčit" tyto vstupy místní aplikaci, tak aby si myslela, že pochází od nějakého místního uživatele, a vzala je na vědomí a vykonala to, co požadují.





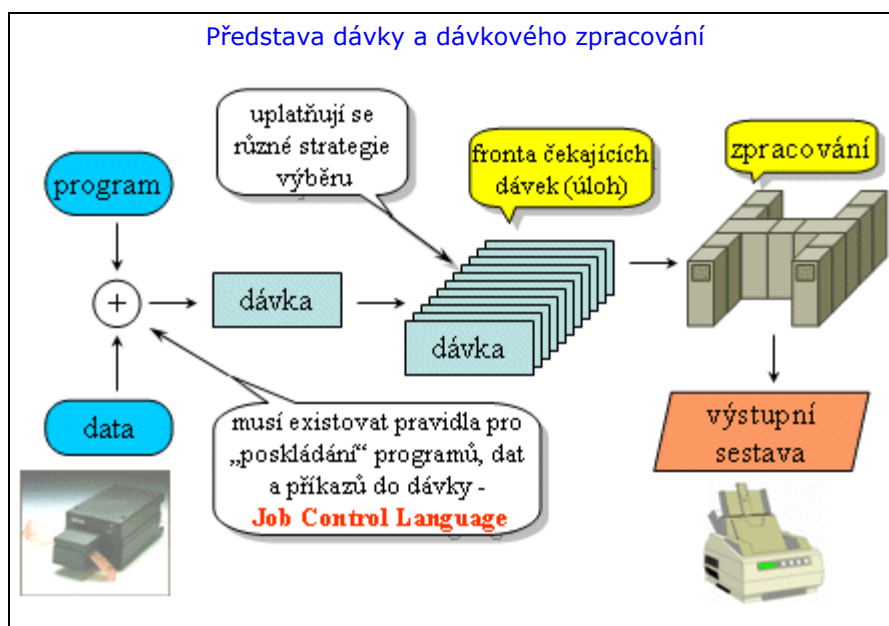
## Výpočetní model

Většina aplikací, se kterými se můžeme setkat v prostředí dnešních počítačových sítí, vychází z výpočetního modelu klient/server. Není to ale zdaleka jediná možnost, protože vedle ní existovala a stále existuje celá řada zajímavých alternativ. Historicky nejstarší bylo tzv. dávkové zpracování, které ani dnes není zdaleka mrtvé. Následoval model host/terminál, po kterém přišly na řadu monolitické aplikace pro izolovaná PC. Jejich nedostatky řešilo propojení počítačů do lokální sítě a využití modelu file server/pracovní stanice. A jak vypadá a funguje 3-úrovňový model klient/server?

### Dávkové zpracování

Zřejmě nejstarším výpočetním modelem je tzv. **dávkové zpracování** (anglicky: batch processing). To vzniklo v době, kdy počítače byly řízeny děrnými štítky či děrnými páskami, a ještě neměly žádné uživatelské terminály (coby pracoviště pro jednotlivé uživatele). Absence takovýchto terminálů pak nutně znamenala, že uživatelé nemohli být v přímém kontaktu se svými programy - nemohli jim zadávat žádné pokyny skrze klávesnici, myš apod., a také samotné programy nemohly svým uživatelům zobrazovat žádné výstupy (přes obrazovku). Takže třeba editace textů na počítači nebyla z principu možná, a žádné textové editory neexistovaly. **Obecně tedy neexistovala možnost interakce**, resp. tzv. interaktivita (mezi uživateli a jejich programy). No a tomu se samozřejmě musel přizpůsobit i tehdejší výpočetní model, tj. dávkové zpracování. To je dodnes antonymem (opakem) k interaktivitě.

**Dávkové zpracování**, které nemá k dispozici interaktivitu, **nutí uživatele** k tomu, **aby všechny své požadavky na zpracování počítačem připravil a přesně vyspecifikoval dopředu**. Tedy aby přesně a jasně popsal, který program má být spuštěn, s jakými vstupními daty má pracovat, kam mají být ukládána výstupní data, generovaná tímto programem, a případně jak se má pokračovat (v případě korektního zakončení programu či v případě chyby). Pokud tak uživatel učiní, a k popisu svých požadavků přidá samotný program i všechna vstupní data, vzniká tím tzv. dávka (anglicky: batch). Tu si lze představit jako určitý celek, který pak může být počítačem proveden (zpracován) v zásadě kdykoli, podle toho, "kdy má čas počítač" (a nikoli podle toho, "kdy má čas uživatel"). Jednotlivé dávky, od stejného uživatele i od dalších uživatelů, se obvykle řadily do fronty, kde čekaly, až na ně dojde řada a budou zpracovány. Uživatel mezitím čekal, třeba i několik hodin, nebo dokonce dnů, než se dočkal nějakého výsledku zpracování své dávky.



Ačkoli dávkové zpracování vzniklo v určité historické době a reagovalo na tehdejší stav výpočetní techniky, není ještě zdaleka mrtvé. I dnešní operační systémy podporují dávky (dávkové soubory), skrze které lze dopředu připravit (naprogramovat) určité činnosti, a pak je jen ve vhodnou dobu spouštět.

Stejně tak se dnes dávkové zpracování používá třeba v souvislosti se superpočítači. Zde si uživatel (na svém osobním počítači) dopředu připraví svůj výpočet i s daty, které chce zpracovat - a pak vše "zabalí" do jedné dávky a tu pošle ke zpracování superpočítači. Už se tomu ale neříká ani tak dávkové zpracování, jako spíše Remote Job Entry (dálkové zadávání úloh), či Remote Job Execution (dálkové zpracování úloh).

### Model host/terminál

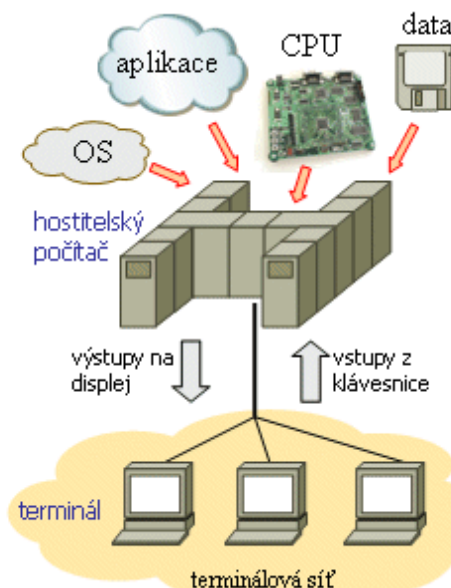
Touhu uživatelů po **přímém kontaktu s jejich aplikacemi** (interaktivitě) bylo možné splnit až v době, kdy se počítače začaly vybavovat vhodnými terminály, či spíše sítěmi terminálů. To proto, aby s počítačem mohlo pracovat více uživatelů. Jenže to nestačilo. **Muselo se vhodně zařídit i to, aby se počítač "věnoval" všem těmto uživatelům současně.** I to bylo novum oproti předchozímu dávkovému zpracování, kdy se počítač celou svou kapacitou věnoval vždy jen jedné úloze.

**Řešení** se nakonec našlo v **technice, označované jako "sdílení času" (time sharing).** Předpokládá, že počítač se vždy po určitou dobu věnuje jednomu uživateli, pak druhému, pak třetímu atd., a stále dokola. Ovšem ony **"určité doby" jsou velmi krátké** (např. desítky milisekund) a **přepínání tak rychlé, že to jednotliví uživatelé ani nemají šanci postřehnout.** Místo toho si mohou myslet, že se celý počítač věnuje právě a pouze jim.

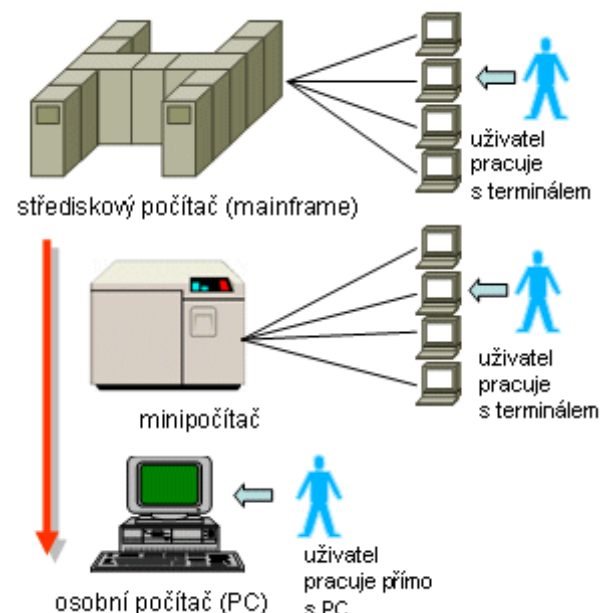
S využitím této techniky již bylo možné vyvinout nový výpočetní model, označovaný jako host/terminál. Pozor ale na to, že slůvko "host" v jeho názvu není odvozeno od českého slova "host" (ve smyslu: být hostem u někoho), ale od anglického "host", což v češtině znamená "hostitel", resp. "být hostitelem někoho, resp. něčeho". Takže je na místě mluvit o hostitelském počítači, který je "hostitelem" pro aplikace, ale i pro data, výpočetní kapacitu a další zdroje.

A právě to je **pro celý model "host/terminál" typické a charakteristické - předpokládá, že veškeré zdroje jsou umístěny "na jedné hromadě" (na hostitelském počítači, a tedy centrálně), a zde se s nimi také pracuje.** K jednotlivým uživatelům, na jejich terminály, pak "putují" již jen výstupy jednotlivých aplikací, zatímco opačným směrem proudí vstupy od uživatelů, určené jejich aplikacím (hlavně vstupy z klávesnic). Pokud vám to připomíná terminálové relace, popisované v minulém dílu tohoto seriálu, v souvislosti s Telnetem, pak vězte že podobnost opravdu není náhodná.

Představa modelu host/terminál



Střediskové počítače, minipočítače a počítače PC



### Monolitické aplikace pro izolovaná PC

Pokrok a vývoj v oblasti hardwaru (i softwaru), který prvním počítačům postupně dopřál terminály (a model host/terminálů), se samozřejmě nezastavil, ale pokračoval dále. Nejprve se ubíral cestou zmenšování hostitelských počítačů, které původně byly opravdu velmi velké. Však se jim také říkalo "střediskové počítače", či převzatě z angličtiny "mainframy". Časem se ale zmenšily do podoby tzv. minipočítačů. Jejich fungování, na

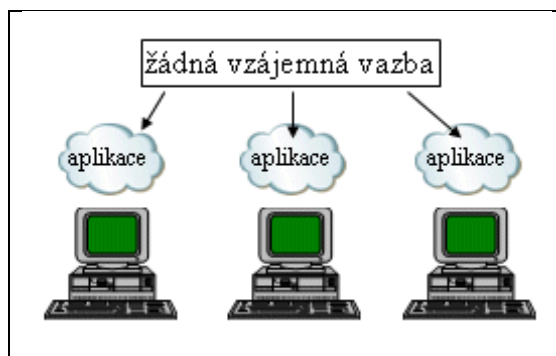
bázi modelu host/terminál, se ale nezměnilo, protože stále musely sloužit více uživatelům současně. A ti s nimi pracovali prostřednictvím terminálů.

Lidé ovšem toužili po tom, aby měli počítač jen a jen ke své dispozici, a nemuseli se o něj dělit s nikým jiným. Na to si museli ještě chvíli počkat, ale nakonec to přišlo také. **Na trh se dostaly osobní počítače (počítače PC, Personal Computer)**, které za svůj přívlastek "osobní" vděčí právě tomu, že už mohou být přiděleny jednomu uživateli do výlučného použití.

Aplikace pro osobní počítače už nemusely počítat s tím, že jejich uživatelé nejsou na hostitelském počítači sami. Nemusely se tedy dělit o dostupné zdroje (výpočetní kapacitu, paměť atd.) s ostatními aplikacemi, které si provozovali ostatní uživatelé, a mohli se na osobním počítači chovat jako jeho výluční páni. Také nemusely čekat na to, ke kterému terminálu si sedne jejich uživatel, aby mu pak posílaly své výstupy právě na tento terminál. Mohly předpokládat, že jejich uživatel s nimi bude komunikovat přes jedinou klávesnici a jedinou obrazovku, kterou je osobní počítač vybaven. Takže měly vlastně o dost snazší pozici, mohly být relativně jednodušší, a mohly být koncipovány jako monolitické (ve smyslu: "v jednom kuse", nikoli nějak dělené).

**Aplikace, běžící na vzájemně izolovaných osobních počítačích, neměly mezi sebou žádnou vazbu.**

Monolitické aplikace, určené pro osobní počítače, dokázaly nabídnout svým uživatelům poměrně vysoký komfort. Měly ovšem také své nevýhody. Například tu, že kdykoli bylo třeba udělat do nich nějaký zásah (například nahrát nějakou aktualizaci, novou verzi, nějaké nové nastavení atd.), muselo se vše dělat na každém počítači, pro každého uživatele. Obecně tedy  $n$ -krát, zatímco dříve, na hostitelském počítači a v rámci modelu



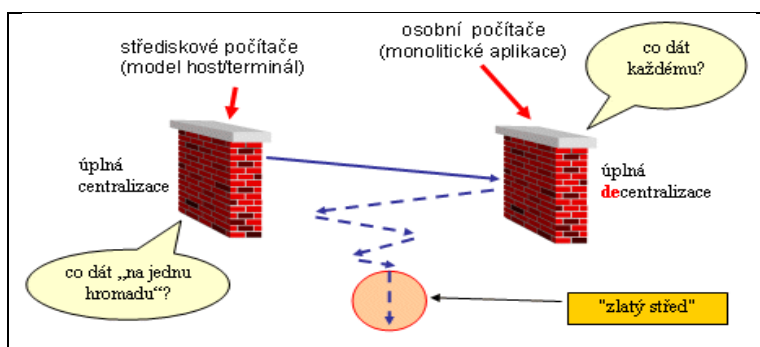
host/terminál, stačilo vše udělat jen 1-krát. Nároky na systémovou správu a údržbu tedy obecně vzrostly  $n$ -krát, což se s postupem času ukázalo jako zásadní problém. Zejména ve firemním prostředí tím výrazně stouply náklady na podporu uživatelů (v rámci tzv. TCO, Total Cost of Ownership), a vynutily si určitý návrat k centralizovaným výpočetním modelům. Tedy k takovým, kde jsou aplikace více soustředěny na nějakém centrálním místě.

**Samostatné a vzájemně izolované počítače však měly** ještě jednu principiální **nevýhodu: neumožňovaly žádné sdílení**. Například když více uživatelů chtělo pracovat se stejnými daty, nešlo to - a každému bylo třeba vytvořit a přidělit jeho vlastní kopii požadovaných dat. Něco takového už nešlo udělat například s drahými periferiemi, jako třeba s laserovými tiskárnami. Dát každému k dispozici kvalitní tiskárnu bylo i v době osobních počítačů zbytečným luxusem, místo kterého obvykle nastupovalo sdílení jedné tiskárny více uživateli.

### Od extrému k extrému, nebo zlatá střední cesta?

**Izolované osobní počítače a jejich monolitické aplikace však nebyly připraveny na žádné sdílení svých zdrojů (včetně periferií).** Samy vlastně představovaly jakýsi přeskok od jednoho extrému k extrému přesně opačnému:

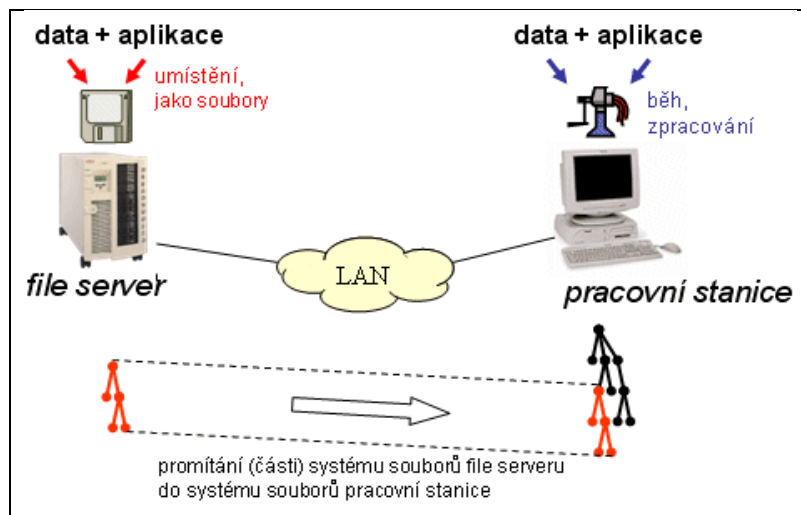
původní výpočetní model host/terminál totiž byl maximálně centralizovaný, kvůli tomu že všechny zdroje u něj byly soustředěny na jednom (centrálním) místě, na samotném hostitelském počítači. Takže sdílení zde bylo velmi snadné. U izolovaných osobních počítačů to ale bylo přesně naopak - všechny byly plně distribuované, resp. decentralizované (rozdělené mezi jednotlivé uživatele, na jejich osobní počítače), a naopak nic nebylo společné. Žádné sdílení zde vlastně ani nešlo realizovat, protože nebylo jak.



Ani jeden extrém není ideální, a hledala se „zlatá střední cesta“. Ta by v daném případě umožňovala sdílet to, co se hodí sdílet, a současně by umožnila přidělit každému do jeho výlučného vlastnictví a používání to, co se naopak sdílet nemusí. Sdílet by se mohly například periférie (již zmiňované tiskárny), nebo třeba datové soubory, ke kterým chce mít přístup více uživatelů. Jenže jak to udělat, když izolované počítače nebyly mezi sebou vůbec propojeny?

### Model file server/pracovní stanice

Prvním krokem k možnosti sdílení bylo vhodné propojení do té doby izolovaných osobních počítačů. K tomu naštěstí již existovala vhodná technologie, vyvinutá právě pro takovýto účel. Ano, byl to Ethernet, nabízející přenosovou rychlost 10 megabitů za sekundu. To bylo dost na to, aby se alespoň sdílení souborů dalo realizovat transparentním způsobem. Tedy tak, aby z pohledu uživatelů a jejich aplikací "nebylo vidět", a soubory, fakticky umístěné na některém jiném počítači v síti, se mohly "tvářit"



a chovat stejně jako soubory lokální, umístěné na daném počítači. Včetně toho, že když s nimi chtěl uživatel (nebo jeho aplikace) pracovat, netrvalo to o nic déle, než u skutečně lokálních souborů.

Jenže pro takovéto plně transparentní sdílení souborů, umístěných na jiném počítači, bylo třeba vymyslet také vhodné technické řešení. Takové, které by dovolilo umístit soubory na jedno centrální místo - na počítač, fungující jako tzv. **file server, resp. souborový server** - ale současně umožnilo ostatním počítačům v síti "vidět je" jako své vlastní soubory. Lze si to představit také tak, že **systém souborů file serveru se "promítá" (tzv. mapuje) do systému souborů jiného počítače (tzv. pracovní stanice).** Jde skutečně jen o iluzi, která ale naplňuje ušlechtilé cíle: díky ní mohou na pracovní stanici (osobním počítači) běžet monolitické aplikace, které vůbec nepočítají s existencí sítě a nějakých vzdálených počítačů (file serverů). Přesto dokáží pracovat s datovými soubory, které jsou ve skutečnosti umístěné na jiném uzlu sítě (ale vůči nim se tváří jako místní).

A dokonce: i tyto samotné aplikace, které o existenci sítě nic netuší, mohou být jako soubory umístěné centrálně, na file serveru, ale "promítat" se do souborového systému pracovní stanice (osobního počítače), a zde být jako aplikace spouštěny a provozovány. Tento jednoduchý trik pak značně zjednodušuje systémovou správu a údržbu těchto aplikací, protože tu lze realizovat na jednom centrálním místě (na file serveru, kde jsou aplikace uloženy jako soubory), místo na každém jednotlivém počítači.

Dodnes se stejné řešení, zde popisované jako "model file server/pracovní stanice", v praxi hojně používá, byť třeba pod úplně jinými jmény. Často se například mluví jen o tzv. **síťových discích (vzdálených discích apod.)**, které plní roli file serveru. Má to i svou logiku: zde popisovaný "model file server / pracovní stanice" vlastně není ani žádným výpočetním modelem. Je zcela záměrně "neviditelný", právě proto aby se mu aplikace nemusely přizpůsobovat. Díky tomu bylo možné hned od začátku používat v prostředí sítě i takové aplikace, které o ní vůbec netušily a myslely si, že pracují na samostatném a izolovaném osobním počítači (který mají navíc samy pro sebe).

## Model klient/server

Model file server/pracovní stanice, popisovaný v předchozím odstavci, tedy umožnil používat v prostředí sítě i takové aplikace, které na to nebyly stavěny. Nebylo to ale zadarmo. Každá taková aplikace, která o existenci sítě netušila, a byla (jako soubor) umístěna

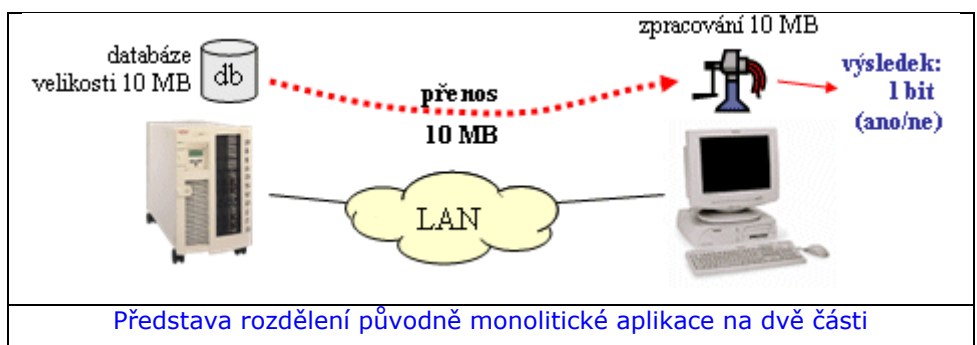


na centrálním file serveru, musela být při svém spuštění nejprve celá přenesena z file serveru na pracovní stanici. Staral se o to systémový software, zajišťující potřebné "mapování" systémů souborů (aniž si to uživatel a jeho aplikace uvědomovali). Obdobně pro datové soubory, se kterými taková aplikace pracovala - také ty se musely přenášet sem a tam, což mohlo výrazně zatěžovat celou lokální síť jako takovou. Jenže to rozhodně nebylo zadarmo. Byla s tím spojena **nemalá reže**.

Zdrojem této reže zde byla skutečnost, že data se zpracovávala jinde, než kde byla umístěna. Pro docenění celého problému si jej můžeme

ukázat na následujícím příkladu, v záměrně zveličené podobě:

*Představme si databázi velikosti například 10 MB, která má být prohledána, zda se v ní nenachází nějaká konkrétní data. Je úplně jedno jaká, ale*



podstatné je, že výsledkem je jednobitová informace: ano, nebo ne. Pokud je ale takováto databáze umístěna (jako soubor) na file serveru, ale díky modelu file server/pracovní stanice bude prohledávána na pracovní stanici, musí být před samotným prohledáním nejprve celá přenesena (z file serveru na pracovní stanici). Přeneseno musí být obecně všech 10 MB dat, které tuto databázi tvoří.

Ted' si ale představme jiný scénář: **databáze se prohledá tam, kde se nachází** (na centrálním file serveru), a **po síti se na pracovní stanici přenesou jen jednobitový výsledek celého prohledání**. Úspora přenosové kapacity je jasná a obrovská. Jenže abychom mohli takovýto scénář realizovat, musíte nejprve **přinutit tu aplikaci, která databázi prohledává, aby se "přestěhovala" z pracovní stanice přímo na centrální file server**. Jenže pak zase nebude moci komunikovat s uživatelem. Takže jinak: **rozdělíme aplikaci napůl, a jedna její část (zajišťující komunikaci s klientem) zůstane na pracovní stanici, zatímco druhá část (ta, která bude skutečně prohledávat databázi), se přestěhuje tam, kde se databáze nachází, tedy na centrální server**. Ale to už jsme dospěli k výpočetnímu modelu klient/server.

Jde o výpočetní model, který již programově počítá s existencí sítě, resp. distribuovaného prostředí, a původně monolitickou aplikaci rozděluje na dvě části: klientskou a serverovou. Pomyslný "řez", který obě části oddělí, se snaží vést tak, aby komunikace mezi klientem a serverem byla co nejméně obsáhlá, tj. aby představovala co nejmenší objemy dat (kvůli zátěži přenosové sítě mezi klientem a serverem).

**Server pak zajišťuje potřebné zpracování, které ale provádí vždy až na žádost ze strany klienta**. Klient zase zajišťuje veškerou komunikaci s uživatelem - přijímá od něj pokyny, a naopak mu zobrazuje (prezentuje) výsledky zpracování, které zajistil server.

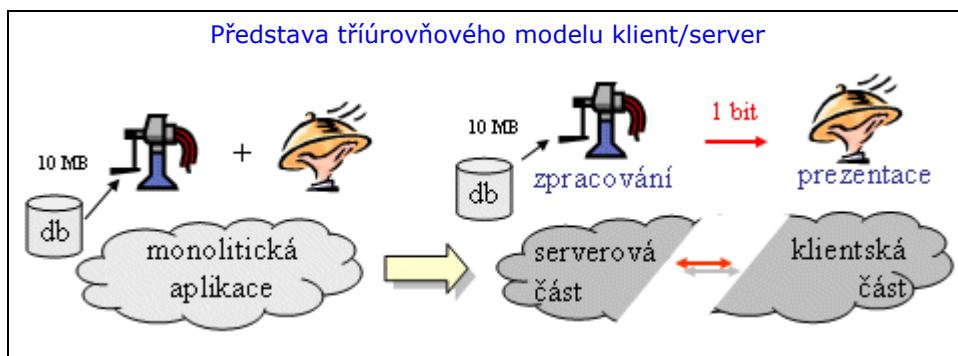
## Tříúrovňový klient/server

Model klient/server je v současné době zřejmě nejrozšířenější výpočetní model, hojně používaný v běžné praxi. To ale zdaleka neznamená, že by neměl žádné nedostatky a nevýhody, a že by od něj neexistovaly "lepší" verze. **Jednou z nevýhod modelu klient/server je to, že i jeho klientská část je "aplikačně závislá", tj. specifická**



**ká pro danou aplikaci.** To na první pohled nevypadá jako nějaká nevýhoda - ale zkusme si představit, že v reálném provozu uživatelé používají několik aplikací, a od každé z nich musí mít na svém počítači nainstalovaného příslušného klienta. A co navíc: o každého takového klienta se musí někdo starat (správce systému). Komplikované to ale je i pro samotného uživatele, protože i on se musí učit pracovat s několika různými klienty, kteří mohou mít různé způsoby a styly ovládání, úplně jinou logiku atd. Nebylo by tedy lepší, kdyby existoval jeden univerzální klient, který by byl využitelný pro více různých aplikací?

**Myšlenka jednoho univerzálního klienta** naštěstí není utopií, jak by se na první pohled mohlo zdát. Dokonce v praxi již existuje a používá se. **Vyžaduje** ale, aby se příslušná aplikace rozdělila nikoli na dvě části, jako u klasického



modelu klient/server, ale hned na tři části. Proto se také mluví o **tříúrovňovém modelu klient/server.**

Možnosti využít jediného univerzálního klienta se dosahuje tím, že se vyčlení (do samostatné části) vše, co je pro danou aplikaci specifické a unikátní. Tím vznikne "prostřední" část, které se ne nadarmo říká **aplikační logika**. Vedle ní se pak již dají použít **dvě univerzální části**:

- **databázový server**, který především uchovává aplikační data, a dokáže v nich vyhledávat (v praxi jde obvykle o běžný SQL server)
- **prezentační část**, která zajišťuje potřebnou "prezentaci" (zobrazování výsledků uživateli, a získávání vstupů od uživatele).

V praxi plní roli **prezentační části** nejčastěji běžný **WWW prohlížeč** (browser). **Aplikační logika** je pak "schována" za příslušným **WWW serverem**, a "z druhé strany" je na ni napojen **databázový server**.

**PŘÍKLAD:** V roli praktického příkladu si představme třeba vyhledávání v jízdním řádu: všechny údaje o existujících spojeních jsou uloženy v databázovém serveru (SQL serveru), který v nich dokáže standardním způsobem vyhledávat (pomocí SQL příkazů). Nerozumí ale tomu, co jednotlivé údaje v databázi znamenají, jak spolu souvisí, jak na sebe jednotlivé spoje navazují atd. Všechny takovéto informace, specifické pro aplikaci spočívající ve vyhledávání spojení, jsou soustředěny v aplikační logice. Ta z jedné strany "rozumí" dotazům uživatelů, a "z druhé strany" rozumí datům, obsaženým v databázi. Dokáže tedy správně interpretovat dotazy uživatelů, hledat na ně odpovědi, a tyto odpovědi pak zase zpětně vrátit uživatelům. Ovšem o prezentaci těchto výsledků už se stará prezentační část, "postavená" na WWW serveru a browseru. Uživatel klade své dotazy skrze webové stránky (kde zadá svůj dotaz v příslušném formuláři), a také odpovědi na své dotazy dostává v podobě webových stránek, které mu zobrazuje jeho prohlížeč. Nepotřebuje tedy žádného specifického klienta pro vyhledávání dopravních spojení, ale pro danou aplikaci (i další aplikace) vystačí jen s jedním klientem - běžným WWW prohlížečem. Navíc může takového univerzálního klienta použít skutečně kdekoli, po celém světě, v dosahu služby WWW.

Na závěr si jen zkusme představit, jak by to dnes vypadalo, pokud by neexistoval takovýto 3-úrovňový model klient/server, a pro každé trochu specializované vyhledávání na Internetu by byl zapotřebí samostatný klient!





# Příkazy pro kontrolu síťové komunikace.

## ping

Program **ping** (anglicky Packet InterNet Groper) umožňuje prověřit funkčnost spojení mezi dvěma síťovými rozhraními (počítače, síťová zařízení) v počítačové síti, která používá rodinu protokolů TCP/IP. Ping při své činnosti periodicky odesílá IP datagramy a očekává odezvu protistrany. Při úspěšném obdržení odpovědi vypíše délku zpoždění (latence) a na závěr statistický souhrn.

### Popis činnosti

Parametrem programu ping je doménové jméno nebo IP adresa síťového rozhraní, jehož dostupnost chceme prověřit. Je-li uvedeno doménové jméno, je nejprve přeloženo pomocí DNS na IP adresu. Program využívá zprávy „Echo Request“ (typ 8, výzva) a „Echo Reply“ (typ 0, odpověď) protokolu ICMP. Výzvy jsou odesílány na cílovou IP adresu a ve stanoveném limitu se očekává odpověď (typicky 3 sekundy). Jednotlivé výzvy obsahují čísla (icmp\_seq), podle kterých je možné identifikovat jednotlivé odpovědi nebo jejich ztrátu. Program průběžně vypisuje, které odpovědi již došly a s jakým zpožděním (latencí).

### Příkaz Ping (Linux)

Parametry

- -i interval s jakým se budou pakety posílat
- -s velikost paketu
- -t TTL paketu (maximální počet skoků k cíli)

Pro popis ostatních parametrů použijte:

```
ping --help
```

### Příklad výpisu

```
[login@localhost ~]$ ping cs.wikipedia.org
PING rr.pmtpa.wikimedia.org (66.230.200.100) 56(84) bytes of data.
64 bytes from rr.pmtpa.wikimedia.org (66.230.200.100): icmp_seq=0 ttl=49 time=193 ms
64 bytes from rr.pmtpa.wikimedia.org (66.230.200.100): icmp_seq=1 ttl=49 time=184 ms
64 bytes from rr.pmtpa.wikimedia.org (66.230.200.100): icmp_seq=2 ttl=49 time=179 ms
64 bytes from rr.pmtpa.wikimedia.org (66.230.200.100): icmp_seq=3 ttl=49 time=181 ms
64 bytes from rr.pmtpa.wikimedia.org (66.230.200.100): icmp_seq=4 ttl=49 time=192 ms
64 bytes from rr.pmtpa.wikimedia.org (66.230.200.100): icmp_seq=5 ttl=49 time=182 ms
64 bytes from rr.pmtpa.wikimedia.org (66.230.200.100): icmp_seq=6 ttl=49 time=176 ms

--- rr.pmtpa.wikimedia.org ping statistics ---
 7 packets transmitted, 7 received, 0% packet loss, time 6003ms
rtt min/avg/max/mdev = 176.643/184.297/193.533/5.889 ms, pipe 2
```

V příkladu byl výpis přerušen kombinací kláves CTRL+c po 7 vyslaných paketech na server

rr.pmtpa.wikimedia.org. Odpovědi se „vrátily“ v průměrném čase 184,3 ms, nejkratší dosažený čas je 176,6 ms a nejdelší je 193,5 ms. Žádný paket nebyl ztracen.

Délku odesílaného paketu je možné nastavit. Délky těsně pod hranici MTU (Maximum Transmission Unit, obvykle 1500 bajtů) umožní sledovat chování sítě při přenosu delších datagramů, což se hodí zejména v bezdrátových sítích, kde kratší datagramy projdou snadněji. Nemá však smysl délku zvětšovat nad velikost MTU, protože dojde k automatické fragmentaci datagramu. Výhodou unixových verzí je, že obvykle kontrolují i obsah vráceného datagramu, takže lze zjistit, jestli nedochází k jejich poškození.

### Příkaz Ping (Windows)

- ping -t – odesílat až do přerušení pomocí kombinace kláves CTRL+c
- ping -l velikost paketu
- ping -i TTL paketu (maximální počet skoků k cíli)

Pro popis ostatních parametrů použijte:

```
ping /? nebo ping ?
```

## Příklad výpisu

```
C:\>ping cs.wikipedia.org
```

Příkaz PING na rr.knams.wikimedia.org [91.198.174.2] s délkou 32 bajtů:

Odpověď od 91.198.174.2: bajty=32 čas=19ms TTL=60

Odpověď od 91.198.174.2: bajty=32 čas=19ms TTL=60

Odpověď od 91.198.174.2: bajty=32 čas=19ms TTL=60

Odpověď od 91.198.174.2: bajty=32 čas=19ms TTL=60

Statistika ping pro 91.198.174.2:

Pakety: Odeslané = 4, Přijaté = 4, Ztracené = 0 (ztráta 0%),

Přibližná doba do přijetí odezvy v milisekundách:

Minimum = 19ms, Maximum = 19ms, Průměr = 19ms

Ve výpisu je vidět, že standardní program **ping v systému Microsoft Windows neumožňuje měřit s přesností větší než 1 milisekunda**. Proto vznikl pro Microsoft Windows nový ping tzv. "True Ping" který měří přesněji a není závislý na konkrétní verzi Windows. **Ping skončí automaticky po 4 pokusech**.

## netstat

anglicky **network statistics** je nástroj pro příkazový řádek, který **zobrazuje aktivní (resp. navázaná) síťová spojení (příchozí i odchozí), směrovací tabulku a řadu statistik síťového rozhraní**. Je dostupný jak na unixových systémech (Linux, BSD), tak v systémech Windows NT. Typicky je používán při hledání problémů v síti nebo ve funkci síťových programů.

### Parametry příkazu:

Přepínače jsou uvozeny pomlčkou (-), nikoli lomítkem (/).

- **-a** zobrazí všechna aktivní spojení a TCP a UDP porty na kterých naslouchají spuštěné procesy
- **-b** zobrazí názvy spustitelných souborů (programů), které mají otevřené spojení nebo naslouchají na síťovém portu (platí pro systémy Windows XP, Windows Server 2003 a novější) pokud se na MacOS-X zkombinuje s přepínačem **-i**, vrátí celkový počet přenesených bajtů
- **-e** zobrazuje ethernetové statistiky odeslaných a přijatých dat v bajtech a paketech; může být zkombinován s **-s**.
- **-f** zobrazuje plná doménová jména (FQDN) (dostupné pouze ve Windows Vista a novějších)
- **-g** zobrazuje informace o členech multicastových skupin pro jak IPv4 tak IPv6 (může být dostupné pouze v novějších operačních systémech)
- **-i** zobrazuje statistiky pro zadané síťové rozhraní (není dostupné ve Windows)
- **-n** IP adresy čísla portů jsou vyjádřena číselně (není použit reverzní převod pomocí DNS na jména)
- **-m** zobrazuje statistiky streamů
- **-o** zobrazuje aktivní TCP spojení včetně ID (PID) procesu příslušného spojení; může být kombinován s přepínači **-a**, **-n** a **-p**; funkční ve Windows XP, Windows 2003 Server (s hotfixem i ve Windows 2000)
- **-p** v systémech Windows a BSD zobrazuje specifikovaný protokol (tcp, udp, icmp, ip, tcpv6, udpv6, icmpv6 nebo ipv6)
- **-p** v Linuxu zobrazuje, které procesy používají které sokety (podobné jako **-b** pod Windows); vyžaduje, aby byl spuštěn s právy správce
- **-p** v systému Solaris zobrazuje specifikovaný protokol (ip, ipv6, icmp, icmpv6, igmp, udp, tcp, nebo rawip)
- **-r** zobrazuje obsah IP routovací tabulky (stejně jako příkaz route print ve Windows)

- -s zobrazuje statistiky protokolu, implicitně jsou zobrazovány pro TCP, UDP, ICMP a IP; pokud je nainstalován protokol IPv6 ve Windows XP, jsou zobrazeny statistiky TCP přes IPv6, UDP přes IPv6, ICMPv6 a IPv6 protokoly; parametr -p může být použit k upřesnění nastavení protokolu
- -v když je použit ve spojení s -b, zobrazí posloupnost komponent zahrnutých ve vytvoření spojení nebo naslouchání portu
- -interval znovu zobrazí vybrané informace dle zadaného intervalu; stisknutí CTRL+C přeruší zobrazení; když je parametr vynechán, netstat vypíše vybrané informace pouze jednou
- -h (unix) /? (windows) zobrazí nápovědu

### Popis výstupu

**Výstup** příkazu netstat obsahuje dvě části. První je věnovaná protokolům TCP a UDP a druhá (která je k dispozici jen v unixových systémech) popisuje unixové sokety.

### Výstup pro TCP/IP

Následující příklad popisuje stav protokolů TCP a UDP (tučně vyznačený text je zadán jako příkaz na příkazovém řádku v Linuxu):

```
$ netstat -pantu
```

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:53	0.0.0.0:*	LISTEN	1257/named
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	1329/sshd
tcp	0	52	10.1.0.1:22	10.1.0.10:57566	SPOJENO	2284/0
udp	0	0	0.0.0.0:53	0.0.0.0:*		1257/named

**Proto** - Sloupec Proto obsahuje název protokolu- Jméno protokolu (TCP nebo UDP).

**Local Address** - Sloupec Local Address (místní adresa) obsahuje IP adresu místního konce spojení, za dvojtečkou pak číslo používaného portu (neurčený port obsahuje hvězdičku). Protože byl zadán přepínač -n, jsou místo jmen zobrazeny IP adresy. Pokud není port doposud nastaven, číslo portu se zobrazí jako hvězdička (\*).

**Foreign Address** - Sloupec Foreign Address (vzdálená adresa) obsahuje IP adresu a port vzdáleného konce spojení (typicky IP adresa vzdáleného počítače).

**State** - Sloupec State (stav) popisuje interní stav TCP spojení (CLOSE\_WAIT, CLOSED, ESTABLISHED, FIN\_WAIT\_1, FIN\_WAIT\_2, LAST\_ACK, LISTEN, SYN\_RECEIVED, SYN\_SEND a TIME\_WAIT).

**PID/Program name** - Sloupec PID/Program name (číslo procesu/název programu) obsahuje identifikaci spuštěného programu, kterému příslušné spojení přísluší.

V následujícím výpisu jsou zobrazeny otevřené unixové sokety:

Active UNIX domain sockets (w/o servers)

Proto	RefCnt	Flags	Type	State	I-Node	Path
unix	11	[ ]	DGRAM		11237	/dev/log
unix	3	[ ]	STREAM	CONNECTED	11848	@/var/run/hald/dbus-LRRvLjNTgb

### Příklady

Zobrazujeme statistiky pouze TCP a UDP protokolů. Napište jeden z následujících příkazů:

```
netstat -sp tcp
```

```
netstat -sp udp
```

zobrazí aktivní TCP spojení a ID procesů každých 5 sekund. Napište následující příkaz (Na Microsoft Windows funguje jedině na XP a 2003 nebo na Windows 2000 s hotfixem):

```
netstat -o 5
```

Mac OS X version

```
netstat -w 5
```

Zobrazuje aktivní TCP spojení a ID procesy pomocí číselné podoby. Napište následující příkaz (Na Microsoft Windows funguje jedině na XP a 2003 nebo na Windows 2000 s hotfixem):

```
netstat -no
```

Zobrazuje všechny otevřené porty a procesy s id 'pid'

```
netstat -ao | grep "pid"
```

## tracert

Program **tracert** slouží k analýze počítačové sítě. Vypisuje uzly (resp. směrovače) na cestě datagramů od zdroje až k zadanému cíli. Uzly jsou zjišťovány pomocí snížení hodnoty TTL v hlavičce datagramů.

Program *tracert* je standardně obsažen ve většině unixových systémů. V systému Microsoft Windows je přímo v základní instalaci obsažen program *tracert*. Pro IPv6 existuje varianta *tracert6*.

### Popis funkce

Tracert zvyšuje hodnotu "time to live" (TTL) po každém úspěšném odeslání balíčku paketů. První tři pakety mají jednotnou hodnotu TTL nastavenou na 1 (odesílají se současně), další tři pakety mají hodnotu TTL 2 atd. Při cestě k cíli paket prochází jednotlivými směrovači (uzly). Při průchodu směrovač sníží hodnotu TTL o 1 a pošle ho dál. Je-li hodnota TTL paketu nula a není v cílové IP síti, pak je paket zahozen a směrovač pošle chybovou ICMP zprávu odesílateli. Tracert využívá právě těchto chybových hlášení, aby sestavil tabulku cesty paketu od odesílatele k cíli. Ve výpisu jsou tak zobrazeny všechny uzly, které položku TTL snižují (tj. směrovače).

Odesílané datagramy mohou být UDP, ICMP nebo TCP. V Microsoft Windows se standardně používá ICMP, v unixových systémech se standardně používá UDP, ale použitý typ lze změnit pomocí parametru (-I, -T, -U).

### Použití

Méně často se používá pro zjišťování problémů se sítí. Díky výpisu jednotlivých uzlů, přes které paket prochází, se zjistí přesná cesta k počítači nebo nějaké stanici v síti. Toto pomáhá identifikovat problémy s routery nebo firewally, které mohou blokovat přístup do sítě.

Tracert lze také použít při stahování dat z více mirrorů, kdy lze trasováním vybrat nejvýhodnější.

### Příklad

Použití příkazu *tracert* v příkazovém řádku Windows:

```
C:\DocsSettings\rawiry>tracert seznam.cz
```

```
Výpis trasy k seznam.cz [77.75.76.3]
```

```
s nejvýše 30 směrováními:
```

1	< 1 ms	< 1 ms	< 1 ms	147.230.163.250
2	< 1 ms	< 1 ms	< 1 ms	147.230.250.65
3	< 1 ms	< 1 ms	< 1 ms	router-h.tul.cz [147.230.250.18]
4	< 1 ms	< 1 ms	< 1 ms	147.230.250.49
5	47 ms	4 ms	203 ms	r84-r40.cesnet.cz [195.113.156.110]
6	4 ms	4 ms	4 ms	nix-pv.pater.iol.cz [194.50.100.160]
7	4 ms	4 ms	4 ms	194.228.21.101
8	4 ms	4 ms	4 ms	194.228.36.1
9	4 ms	4 ms	4 ms	www.seznam.cz [77.75.76.3]

```
Trasování bylo dokončeno.
```

### Použití v Linuxu:

```
[huzva@pluto ~]$ traceroute idnes.cz
```

```
traceroute to idnes.cz (194.79.52.192), 30 hops max, 40 byte packets
```

```

1 neptun.domena.cz (10.200.0.1) 0.188 ms 0.105 ms 0.095 ms
2 ten155-r1-nextel.cesnet.cz (195.178.64.117) 0.530 ms 0.691 ms 0.467 ms
3 r84-r40.cesnet.cz (195.113.156.110) 23.732 ms 23.817 ms 24.205 ms
4 nix1.mafra.cz (194.50.100.156) 4.268 ms 4.261 ms 4.264 ms
5 194.79.55.3 (194.79.55.3) 4.360 ms 4.737 ms 4.883 ms

```

V příkladech je vidět, že v Linuxu se zobrazuje výstup s vyšší přesností.

## Bezpečnost

Traceroute zpřístupňuje velmi detailní informace o jednotlivých bodech na cestě k nějakému cíli v síti. Na počátcích používání internetu to bylo považováno za přijatelné, ale s následujícími problémy to vyvolalo debatu ohledně bezpečnosti a ochrany soukromých informací. **Traceroute začali totiž hojně zneužívat hackeři.** Získávali tak podstatné informace o síťové architektuře různých společností. Pomocí použití příkazu *traceroute* mohli hackeři rychle zmapovat uzly, které daná společnost měla k dispozici, a určit si slabý bod, který mohli prolomit.

Kvůli těmto důvodům mnoho správců sítí **zablokovalo odesílání odpovědí programu traceroute ze svých routerů, proto některé uzly na cestě neodpovídají a trasování často končí na hranici lokální sítě.**

## Přehled příkazů s typickými parametry:

- **ping** - testuje spojení se vzdáleným PC, měří dobu odezvy (latenci)
  - ping 192.168.10.5 - cílový PC podle IP
  - ping -t komp1 - cílový PC podle DNS jména (provede se překlad) a neskončí po 4 paketech
- **ipconfig** - konfigurace síťových adaptérů (zobrazení TCP/IP hodnot)
  - ipconfig /all - podrobný výpis
  - ipconfig /renew - obnoví IP adresu rozhraní
  - ipconfig /registerdns - obnoví DHCP pronájem a znovu zaregistruje adresu u DNS
  - ipconfig /flushdns - vyprázdní DNS cache
  - ipconfig /displaydns - zobrazí DNS cache
- **nslookup** - nalezení DNS záznamů, zjednodušeně IP adresu k doménovému jménu, po spuštění má vlastní prompt: nslookup www.seznam.cz - vrátí A záznam z primárního DNS pro zadanou doménu
  - ls -d {domena} - výpis všech záznamů (pokud mám oprávnění) = Zone Transfer
- **arp** - práce s ARP tabulkou (mapování IP adres na MAC adresy)
  - arp -a - vypíše ARP tabulku
  - arp -d 157.55.85.212 - smaže záznam
  - arp -s 157.55.85.212 00-aa-00-62-c6-09 - vloží záznam
- **netstat** - aktivní TCP spojení na portu
  - netstat -b - vypíše aplikace, které vytvořily spojení
  - netstat -r - vypíše routovací tabulku
  - netstat -ano | findstr 3044 - hledání
- **netdiag** - informace o připojení klienta, diagnostický nástroj, který testuje připojení počítače do sítě, ověřuje autorizaci apod
- **netsh** - network shell - konfigurace sítě, po spuštění má vlastní prompt, řádkové příkazy (shell) pro konfiguraci všeho ohledně sítě, například netsh interface ip konfigurace IP adresy
- **tracert** - funkce traceroute, sleduje cestu k cíli (přes jaké uzly/hopy)
  - tracert www.google.com - pinguje jednotlivé hopy
- **net** - široká paleta příkazů okolo sítí, sdílení, účtů, služeb
  - net use User heslo - změna hesla lokálního uživatele User na zadané heslo
  - net use - zobrazí mapované sharey
  - net use u: \\ok\d - namapuje síťovou cestu na disk

- net user uzivatel - zobrazí informace o lokálním účtu
  - net user uzivatel /expire:1.10.2011 - pro lokální účet nastaví dobu platnosti
  - net help user - nápověda k příkazu net user
- **route** - informace o routovací tabulce
  - route print - vypíše routovací tabulku
- **telnet** - klient pro službu telnet, vhodný pro testování běhu aplikace na nějakém portu
  - telnet www.google.com 80 - připojí se na danou adresu a port
- **ftp** - jednoduchý klient pro FTP
- **tftp** - jednoduchý klient pro TFTP
- **pathping** - kombinace funkcí ping a traceroute, rychle projde cestu k cíli a vypíše hopy, pro každý hop provede statistiku pomocí pingu
  - pathping -q 10 www.google.com - dotaz na cíl podle DNS, 10 dotazů pro každý hop (zkrátí se délka provedení)
- **nbtstat** - práce s NetBIOS, hlavně pro řešení problémů
  - nbtstat -n - lokálně registrovaná jména
  - nbtstat -c - zobrazí obsah NetBIOS cache
  - nbtstat -RR - obnoví záznam na WINS serveru



# Bezpečnost počítačových sítí

## Způsoby útoků:

- **Pasivní odposlech**

Odposlechnutí veškeré komunikace, která je dostupná. Síťová karta se přepne do tzv. promiskuitního režimu, kdy přijímá veškeré pakety i ty které jí vůbec nenáleží. Pokud nepoužíváme šifrovanou komunikaci při autentizaci (zalogování) lze snadno odposlechnout uživ. jméno a heslo. (Praktické cvičení program Ethereal.)

- **Vkládání falešných informací**

Útočník se paralelně připojí k vysílacímu kanálu a vkládá do něj vlastní datové bloky. Je to např. vysílání poštovních zpráv pod falešnou poštovní adresou. Zde dochází k narušení integrity dat.

- **Útok pomocí elektronické pošty**

Pomocí e-mailu nám může někdo spolu s dopisem poslat i soubory dat a spustitelné programy, které nám pak mohou zničit data na disku. Je totiž zcela jednoduchou záležitostí zjistit si někčí e-mail adresu.

- **Fyzické přerušení komunikační cesty**

1. **Zahlcení** - přetížení komunikačního kanálu vytrvalým vysíláním nesmyslných zpráv. Provádí se posíláním nesmyslných požadavků na serveru. V praxi se často používá přes ICMP protokol, který nazýváme DOS útok.
2. **Změna komunikační trasy** - změna routovacích informací komunikačního subsystému.
3. **Přerušení komunikace** - Tímto útokem jsou nejvíce ohroženy především ty komunikační sítě, které nemají vyřešen náhradní způsob komunikace pomocí alternativních komunikačních kanálů.
4. **Zdržení a zpoždění informace** - Zadržetí a zpoždění informace během přenosu je nebezpečné především pro systémy, které pracují v reálném čase.

**Bezpečnost bezdrátových sítí** je v poslední době také velmi diskutovaným tématem. Napadení nezabezpečené nebo nedostatečně zabezpečené bezdrátové sítě je z hlediska útočníka velmi jednoduché, neboť pro připojení k síti nepotřebuje fyzický přístup k síťové infrastruktuře. Zabezpečení bezdrátových sítí představuje mimořádně důležitý a velmi často podceňovaný prvek firemní bezpečnosti.

## Typy útoků na bezdrátovou síť:

### Prolomení klíče WEP

**Falešná MAC adresa** – Některá zařízení umožňují kontrolovat přístup do bezdrátové sítě pomocí jedinečné hardwarové adresy zařízení. Přístupový bod pak dovoluje navázat spojení pouze zařízením, jejichž MAC adresa je na seznamu povolených zařízení.

**Útok Man-in-the-Middle** – Cílem útočníka je včlenit se mezi účastníka a přístupový bod a odposlouchávat veškerý provoz mezi nimi.

**Podvržené spojení** – Útočník zprovozní vlastní přístupový bod, který předstírá, že je součástí sítě. Na ten se pak účastníci připojují se svými přístupovými údaji (přihlašovací jméno a heslo), které útočník takto získá.

**Denial of Service (DoS)** – Zaplavení přístupového bodu množstvím zpráv, které znemožní účastníkům přístup na tento bod.

## Obrana před útoky:

### Zabezpečení na hardwarové úrovni

1. **Ochrana komunikačních portů**

- Aktivní prvek sítě má ve své databázi uložena čísla komunikačních portů, vypnutí portu, které v současné době nejsou potřebné pro komunikaci

## 2. Zabezpečení kabelových spojů

- musí být zajištěno, aby se kdokoli nemohl neoprávněně připojit
- kabely se proto ukládají do zdi trubek lišt apod.

## 3. Řízení směru toku dat

- použití vhodných aktivních síťových prvků mostů, routerů, switchů
- nastavíme komunikační trasy pro chráněná data, povolíme trasy důvěryhodné a zakážeme nebezpečné
- filtrace na základě adresy příjemce, odesílatele a typu paketu

## 4. Zabezpečení serveru

- server by měl být umístěn v samostatné uzamykatelné a klimatizované místnosti
- zabezpečení přihlašování, zakázat „jednoduchá“ hesla, pravidelná změna hesel
- přihlašování lze též řešit hardwarovými klíči, otisky prstů, skanování sítnice apod.

# Gateway (brána)

**Gateway** (brána) je v počítačových sítích uzel, který spojuje dvě sítě s odlišnými protokoly. Brána musí vykonávat i funkci routeru (směrovače), a proto ji řadíme v posloupnosti síťových zařízení výše. Brána například přijme z Internetu pomocí webové stránky zprávu, kterou odešle do mobilní GSM sítě v podobě SMS zprávy. Pojem **default gateway** (implicitní brána) označuje router (směrovač), přes který se stanice dostanou do vnější sítě (tj. obvykle do Internetu). Oba významy jsou často nesprávně zaměňovány.

## Funkce brány

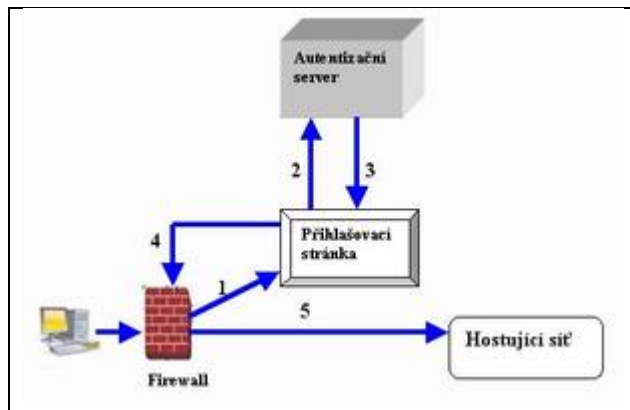
**První typem brány** je brána pracující na aplikační úrovni (viz Referenční model ISO/OSI). Kromě výše zmíněného příkladu propojení GSM sítě a Internetu to jsou i brány mezi různými sítěmi pro zasílání zpráv (ICQ, Jabber, ...) a podobně. Brána přijme celou zprávu, která se může skládat z mnoha menších částí (např. datagramů). Pak zprávu převede do formátu určeného pro cílovou síť a odešle. Bránu tak tvoří speciální program spuštěný na počítači, který je připojen do obou různých sítí (tj. například do Internetu pomocí síťové karty a do GSM sítě pomocí mobilního telefonu připojeného přes sériový port).

**Druhým typem brány** je brána pracující na transportní nebo síťové vrstvě. Tyto brány pracují na nižší síťové vrstvě, například přímo s datagramy. V tomto případě brána nedekóduje celou zprávu, ale jen transformuje datagramy jedné sítě do datagramů sítě druhé. Příkladem je tzv. SOCKS, kde dochází k přenosu datagramů TCP/IP přes síť, která TCP/IP nepodporuje. V takovém případě TCP/IP aplikace používá speciální subsystém, který zajistí doručení datagramů až k bráně pomocí jiného protokolu (např. IPX/SPX) a v bráně probíhá zpětný převod do protokolu TCP/IP a odeslání do cílového místa v Internetu. Převod pracuje samozřejmě oběma směry.

## Povolení přístupu pomocí brány

Autentizace přes webové rozhraní:

1. klient je vyzván k zadání přihlašovacích údajů
2. stránka odešle údaje na autentizační server
3. autentizační server odpoví, zda je uživatel známý nebo ne
4. pro oprávněného uživatele je firewall otevřen
5. klient bez omezení komunikuje



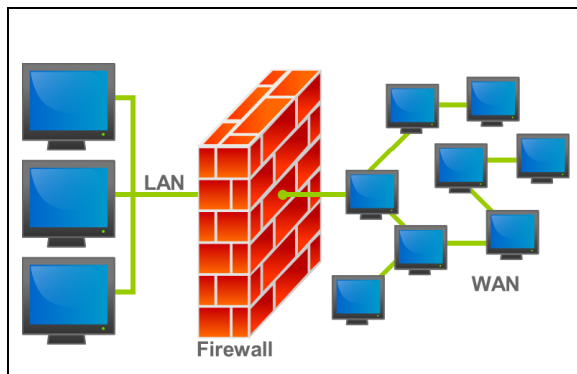
Některé počítačové sítě se chrání proti přímému přístupu cizích počítačů. Přinese-li si někdo do takové sítě notebook, musel by si nejprve zdůvodnit administrativní povolení, svůj počítač složitě konfigurovat (viz

IEEE 802.1X), instalovat doplňující software a podobně. Z tohoto důvodu může být přístup ke zdrojům takové sítě omezen pomocí speciální brány.

Připojí-li se do sítě neznámý počítač (neznámá IP adresa, MAC adresa a podobně), je veškerý jeho provoz přesměrován na bránu (např. pomocí VLAN). Ve spolupráci s webovým prohlížečem, který je dnes součástí výbavy většiny počítačů, umožní brána na základě znalosti uživatelského jména a hesla autentizaci. Po úspěšné autentizaci zajistí brána rekonfiguraci síťových prvků, které odblokují přístup připojeného počítače.

## Firewall

**Firewall** je síťové zařízení hlídající provoz mezi lokální sítí (LAN) a „zbytkem světa“ (WAN), které slouží k řízení a zabezpečování síťového provozu mezi sítěmi s různou úrovní důvěryhodnosti a zabezpečení. Zjednodušeně se dá říct, že slouží jako **kontrolní bod**, který **definuje pravidla pro komunikaci mezi sítěmi, které od sebe odděluje**. Tato pravidla historicky vždy zahrnovala identifikaci zdroje a cíle dat (zdrojovou a cílovou IP adresu) a zdrojový a cílový port, což je však pro dnešní firewally už poměrně nedostatečné – modernější firewally se opírají přinejmenším o informace o stavu spojení, znalost kontrolovaných protokolů a případně prvky IDS.



Firewally se během svého vývoje řadily zhruba do **následujících kategorií**:

- **Paketové filtry**
- **Aplikační brány**
- **Stavové paketové filtry**
- **Stavové paketové filtry s kontrolou známých protokolů a popř. kombinované s IDS**

### Paketové filtry

Nejjednodušší a nejstarší forma firewallování, která spočívá v tom, že pravidla přesně uvádějí, z jaké adresy a portu na jakou adresu a port může být doručen procházející paket, tj. **kontrola se provádí na třetí a čtvrté vrstvě modelu síťové komunikace OSI**.

**Výhodou** tohoto řešení je vysoká rychlost zpracování, proto se ještě i dnes používají na místech, kde není potřebná přesnost nebo důkladnější analýza procházejících dat, ale spíše jde o vysokorychlostní přenosy velkých množství dat.

**Nevýhodou** je nízká úroveň kontroly procházejících spojení, která zejména u složitějších protokolů (např. FTP, video/audio streaming, RPC apod.) nejen nedostačuje ke kontrole vlastního spojení, ale pro umožnění takového spojení vyžaduje otevřít i porty a směry spojení, které mohou být využity jinými protokoly, než bezpečnostní správce zamýšlel povolit.

### Aplikační brány

Jen o málo později, než jednoduché paketové filtry, byly postaveny firewally, které na rozdíl od paketových filtrů **zcela oddělily sítě, mezi které byly postaveny**. Říká se jim většinou *Aplikační brány*, někdy také *Proxy firewally*. Veškerá komunikace přes aplikační bránu probíhá formou dvou spojení – klient (iniciátor spojení) se připojí na aplikační bránu (proxy), ta příchozí spojení zpracuje a na základě požadavku klienta **otevře nové spojení k serveru, kde klientem je aplikační brána**. Data, která aplikační brána dostane od serveru, pak zase v původním spojení předá klientovi. **Kontrola se provádí na sedmé (aplikační) vrstvě síťového modelu OSI** (proto se těmto firewallům říká aplikační brány).

Jedním vedlejším efektem použití aplikační brány je, že server nevidí zdrojovou adresu klienta, který je původcem požadavku, ale jako zdroj požadavku je uvedena vnější adresa aplikační brány. Aplikační brány díky tomu automaticky působí jako nástroje pro překlad adres (NAT), nicméně tuto funkcionalitu má i většina paketových filtrů.

**Výhodou** tohoto řešení je poměrně vysoké zabezpečení známých protokolů.

**Nevýhodou** je zejména vysoká náročnost na použitý HW – aplikační brány jsou schopny zpracovat mnohonásobně nižší množství spojení a rychlosti, než paketové filtry a mají mnohem vyšší latenci. Každý protokol vyžaduje napsání specializované proxy, nebo využití tzv. generické proxy, která ale není o nic bezpečnější, než využití paketového filtru. Většina aplikačních bran proto uměla kontrolovat jen několik málo protokolů (obvykle kolem deseti). Původní aplikační brány navíc vyžadovaly, aby klient uměl s aplikační branou komunikovat a neuměly dost dobře chránit svůj vlastní operační systém; tyto nedostatky se postupně odstraňovaly, ale po nástupu stavových paketových filtrů se vývoj většiny aplikačních bran postupně zastavil a ty přeživší se dnes používají už jen ve velmi specializovaných nasazeních.

### **Stavové paketové filtry**

Stavové paketové filtry provádějí kontrolu stejně jako jednoduché paketové filtry, navíc si však ukládají informace o povolených spojeních, které pak mohou využít při rozhodování, zda procházející pakety patří do již povoleného spojení a mohou být propuštěny, nebo zda musí znovu projít rozhodovacím procesem. To má dvě výhody – jednak se tak urychluje zpracování paketů již povolených spojení, jednak lze v pravidlech pro firewall uvádět jen směr navázání spojení a firewall bude samostatně schopen povolit i odpovědní pakety a u známých protokolů i další spojení, která daný protokol používá.

*Například pro FTP tedy stačí nastavit pravidlo, ve kterém povolíte klientu připojení na server pomocí FTP a protože se jedná o známý protokol, firewall sám povolí navázání řídicího spojení z klienta na port 21 serveru, odpovědi z portu 21 serveru na klientem použitý zdrojový port a po příkazu, který vyžaduje přenos dat, povolí navázání datového spojení z portu 20 serveru na klienta na port, který si klient se serverem dohodl v rámci řídicího spojení a pochopitelně i odpovědní pakety z klienta zpět na port 20 serveru.*

Zásadním vylepšením je i možnost vytváření tzv. virtuálního stavu spojení pro bezstavové protokoly, jako např. UDP a ICMP.

K největším výhodám stavových paketových filtrů patří jejich vysoká rychlost, poměrně slušná úroveň zabezpečení a ve srovnání s výše zmíněnými aplikačními branami a jednoduchými paketovými filtry řádově mnohonásobně snazší konfigurace – a díky zjednodušení konfigurace i nižší pravděpodobnost chybného nastavení pravidel obsluhou.

**Nevýhodou** je obecně nižší bezpečnost, než poskytují aplikační brány.

### **Stavové paketové filtry s kontrolou protokolů a IDS**

Moderní stavové paketové filtry kromě informací o stavu spojení a schopnosti dynamicky otevírat porty pro různá řídicí a datová spojení složitějších známých protokolů implementují něco, co se v marketingové terminologii různých společností nazývá nejčastěji *Deep Inspection* nebo *Application Intelligence*. Znamená to, že firewally jsou schopny kontrolovat procházející spojení až na úrovni korektnosti procházejících dat známých protokolů i aplikací. Mohou tak například zakázat průchod http spojení, v němž objeví indikátory, že se nejedná o požadavek na WWW server, ale tunelování jiného protokolu, což často využívají klienti P2P sítě (ICQ, gnutella, napster, apod.), nebo když data v hlavičce e-mailu nesplňují požadavky RFC apod.

Nejnověji se do firewallů integrují tzv. *in-line IDS (Intrusion Detection Systems* – systémy pro detekci útoků). Tyto systémy pracují podobně jako antiviry a pomocí databáze signatur a heuristické analýzy jsou schopny odhalit vzorce útoků i ve zdánlivě nesouvisejících pokusech o spojení, např. skenování adresního rozsahu, rozsahu portů, známé signatury útoků uvnitř povolených spojení apod.

**Výhodou** těchto systémů je vysoká úroveň bezpečnosti kontroly procházejících protokolů při zachování relativně snadné konfigurace, poměrně vysoká rychlost kontroly ve srovnání s aplikačními branami, nicméně je znát významné zpomalení (zhruba o třetinu až polovinu) proti stavovým paketovým filtrům.

**Nevýhodou** je zejména to, že z hlediska bezpečnosti designu je základním pravidlem bezpečnosti udržovat bezpečnostní systémy co nejjednodušší a nejmenší. Tyto typy firewallů integrují obrovské množství funkcionality a zvyšují tak pravděpodobnost, že v některé části jejich kódu bude zneužitelná chyba, která povede ke kompromitování celého systému.

### **Bezpečnostní politika**

Nastavení pravidel pro komunikaci přes firewall se běžně označuje termínem „bezpečnostní politika firewallu“, zkráceně „bezpečnostní politika“. Bezpečnostní politika zahrnuje nejen samotná pravidla komunikace mezi síťmi, ale u většiny dnešních produktů také různá globální nastavení, překlady adres (NAT), instrukce pro vytváření šifrovaných spojení mezi šifrovacími branami (VPN – Virtual Private Networks), vyhledávání možných útoků a protokolových anomálií (IDS – Intrusion Detection Systems), autentizaci a někdy i autorizaci uživatelů a správu šířky přenosového pásma (bandwidth management).

## **PROXY Server**

**Proxy server** funguje jako prostředník mezi klientem a cílovým počítačem (serverem), překládá klientské požadavky a vůči cílovému počítači vystupuje sám jako klient. Přijatou odpověď následně odesílá zpět na klienta. Může se jednat jak o specializovaný hardware, tak o software provozovaný na běžném počítači. Proxy server odděluje lokální počítačovou síť (intranet) od Internetu.

**Aplikační proxy** je server speciálně určený pro určitý protokol nebo aplikaci. Proxy server může analyzovat obsah komunikace, případně ji pozměňovat (např. odstraňování reklam z http požadavků, blokování webových stránek podle obsahu a podobně) nebo ukládat požadavky do vyrovnávací paměti (cache), ze které mohou být při opakovaném požadavku odpovědi poskytnuty rychleji.

### **Typická použití:**

#### **Ochrana soukromí**

Pro cílový server je klientem proxy server a nikoliv původní klient. To má za následek, že cílovému serveru není známa IP adresa původního klienta. Zejména u webových proxy toto opatření není stoprocentní, protože některé z nich adresu klienta přidávají do upraveného požadavku. Úpravou požadavku lze ale zvýšit soukromí ještě víc, a sice odstraňováním cookies nebo jiných informací (např. HTTP referrer – informace o poslední navštívené stránce).

#### **Zvýšení výkonu komunikace**

Pokud se některé požadavky klienta opakují (např. požadavek na stažení loga Wikipedie, dotazy na DNS, atd.), může si proxy server uložit odpověď do vyrovnávací paměti a odpověď odeslat klientovi přímo, aniž by předal komunikaci k cílovému serveru.

#### **Bezpečnost**

Aplikační proxy server může analyzovat komunikaci a zjišťovat přítomnost např. virů. Dále může procházející požadavky šifrovat a dešifrovat.

#### **Připojení více klientů k internetu**

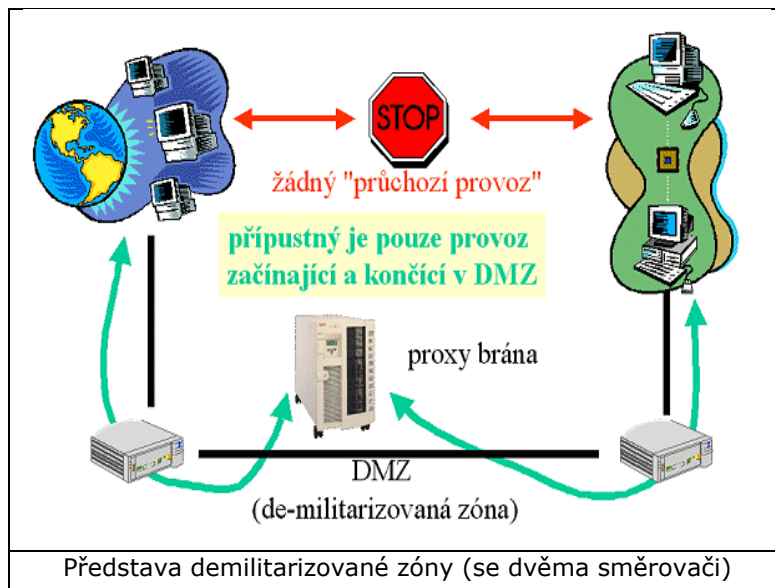
Klienti nemusí mít přiřazeny veřejné IP adresy a přesto, mohou mít přes proxy server přístup ke službám na internetu (toho je také možno docílit překladem IP adres, tzv. NAT, který je často zkombinován s firewalllem). Právě popsané fungování proxy bran i celých firewallů využívajících tyto proxy brány přináší jednu významnou odlišnost oproti řešením, která proxy brány nepoužívají. Jde o to, že existence proxy bran již nemůže být transparentní, resp. nemůže zůstat utajena koncovým uzlům chráněné sítě a aplikacím, které zde jsou provozovány. Například právě WWW browser, jehož uživatel si klikne na nějaký hypertextový odkaz, by správně měl poslat příslušný požadavek přímo tomu serveru, který je v odkazu obsažen. Místo toho ale WWW browser

musí zaslat tento požadavek proxy bráně, která pak teprve zajistí vše potřebné. V praxi se toto řeší nastavením konfigurace WWW browseru, kterému se řekne, že musí spolupracovat s příslušnou proxy bránou. Snad nejdůležitější je ale to, že takovéto nastavení může zůstat skryto před uživatelem - tj. stačí, když jej provede správce sítě, zatímco uživatel o tom nemusí mít ani potuchy a svůj browser používá přesně stejně jako bez firewallu a proxy bran.

## Demilitarizovaná zóna

Druhým typickým provedením firewallu je takové řešení, pro které se vžil poněkud zvláštní označení "demilitarizovaná zóna".

Co takováto demilitarizovaná zóna obnáší v praxi je, že skrze demilitarizovanou nemožnou prostupovat žádné síťové pakety (tj. platí zde zákaz IP forwardingu). Uvnitř demilitarizované zóny pak jsou jednotlivé proxy brány, specializované na konkrétní služby a zprostředkovávající komunikaci mezi propojenými sítěmi.



Představa demilitarizované zóny (se dvěma směrovači)

Pokud jde o možnou realizaci takto fungující demilitarizované zóny, pak zde existuje více možností. Principiálně nejjednodušší je řešení se dvěma směrovači (viz obrázek ...), mezi kterými vzniká příslušná demilitarizovaná zóna jako samostatná síť. Důležité zde je to, aby směrovače byly nakonfigurovány takovým způsobem, aby umožňovaly přenos síťových paketů začínajících či končících v demilitarizované zóně, a naopak zakazovaly veškerý přenos síťových paketů skrz demilitarizovanou zónu. K tomu stačí i naprosto minimální inteligence, kterou mají úplně všechny směrovače.

**Výhodou** právě popsaného řešení se dvěma směrovači je přeci jen větší bezpečnost (případný narušitel musí překonávat dva omezující uzly). **Ne-výhodou** je ale nutnost použít směrovače dva. Existuje však i možnost vystačit s jediným směrovačem, pokud má alespoň tři síťová rozhraní.

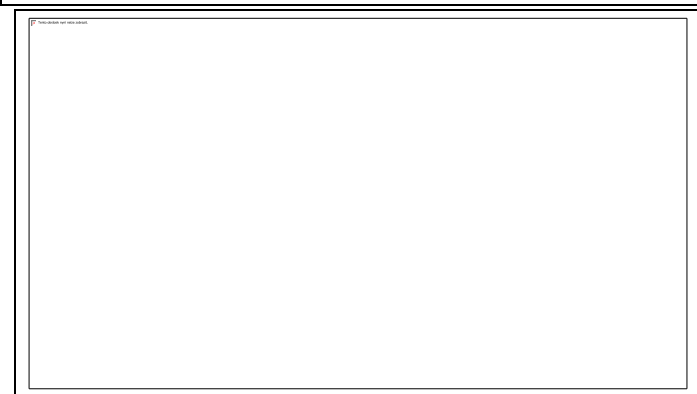
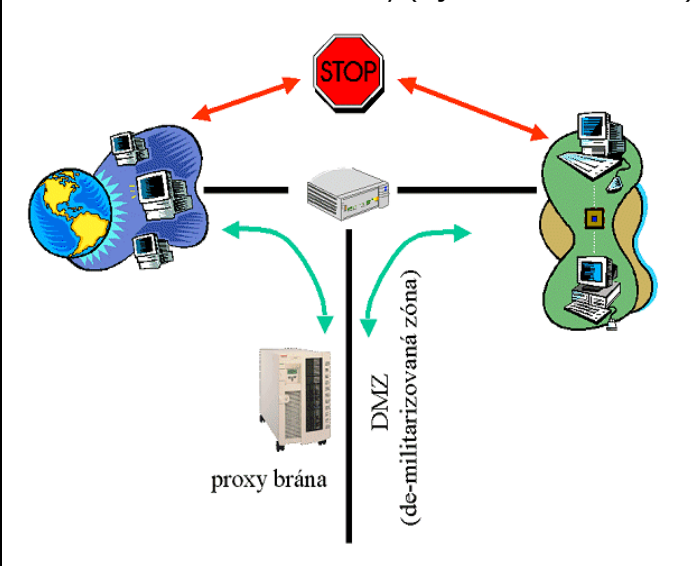
Mezi oběma základními variantami firewallu ("dvounohého" a na principu demilitarizované zóny) samozřejmě existuje celá řada přechodových variant.

Například demilitarizovaná zóna je v tradičním pojetí skutečně samostatnou sítí, do které může být umístěno více různých uzlů, plnících roli proxy bran i další role - například roli WWW serveru, který využívá skutečnosti, že je "vidět" (je dostupný) jak současně uživatele obou propojených sítí, tedy i z chráněné privátní sítě i z veřejného Internetu.

Další možností je eliminování jednoho ze dvou směrovačů (na straně chráněné privátní sítě), a přímé připojení uzlů vnitřní sítě na uzel plnící roli demilitarizované zóny.

Ještě další možností je pak eliminace zbývajících

Představa demilitarizované zóny (s jedním směrovačem)





směrovače, což je vlastně již výše uvažovanou variantou "dvounohého" firewallu. V praxi však může být demilitarizovaná zóna redukována z celé sítě na jeden jediný "dvounohý" počítač, který realizuje všechny potřebné činnosti typické pro demilitarizovanou zónu.

Dnes je tato varianta typická zejména pro připojování menších až velmi malých sítí, včetně sítí domácích, a její velkou předností je, že nahrazuje potřebu samostatného směrovače, který by jinak byl zapotřebí pro připojení k Internetu.

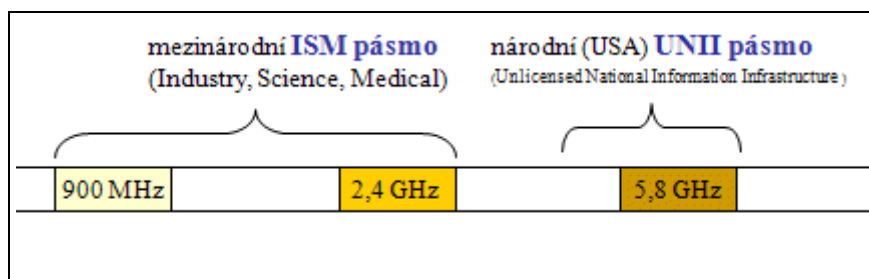
# BEZDRÁTOVÉ SÍTĚ

Vývoj Ethernetu se neubíral pouze cestou zrychlování a zvětšování dosahu „po drátech“, ať již po kroucené dvoulince či po optických vláknech. Časem zákonitě došlo i na myšlenku, zda to nezkusit bez drátů. A to i bez jména „Ethernet“, kterého se bezdrátová verze časem zbavila. Dnes o ní nejčastěji hovoříme jako o technologii Wi-Fi, i když ani to není úplně přesně. Wi-Fi je totiž jen něco jako nálepka, stvrzující že určité řešení plně vyhovuje příslušným standardům a je kompatibilní s dalšími produkty, které vychází ze stejných standardů.

Impulsů, které otevřely cestu k bezdrátovému Ethernetu, bylo určitě více. Kromě snahy zbavit se závislosti na drátech to určitě bylo i rozhodnutí amerického regulátora (FCC, Federal Communications Commission) z roku 1985, kterým tento úřad uvolnil tři „odpadková“ frekvenční pásma pro bezlicenční využití. To byl významný průlom do tehdejší praxe, která (snad kromě radioamatérských pásem) počítala jen s licenčním principem využití frekvenčního spektra. Tedy s tím, že pokud chtěl někdo na nějakých frekvencích něco vysílat, musel k tomu mít individuální oprávnění (individuální licenci). FCC však tehdy zvolil bezlicenční princip, v rámci kterého individuální licence nejsou potřeba a každý, kdo chce určité frekvence využít, tak může učinit - a nepotřebuje k tomu žádnou individuální licenci.

Samozřejmě to neznamena, že si každý může v takovýchto bezlicenčních pásmech dělat co chce. I pro pásma, využívaná na bezlicenční principu, vždy existují určitá pravidla, která říkají co a jak se v nich smí a co naopak nesmí. Například, jaké vysílací výkony nesmí být překročeny. Nebo když dojde k rušení, že musí přestat vysílat ten, kdo začal jako poslední. Všechna takováto pravidla pak jsou zakotvena v „generálních licencích“, které vydává správce kmitočtového spektra a které nejsou adresné - nemají charakter individuálního povolení, ale povolení generálního.

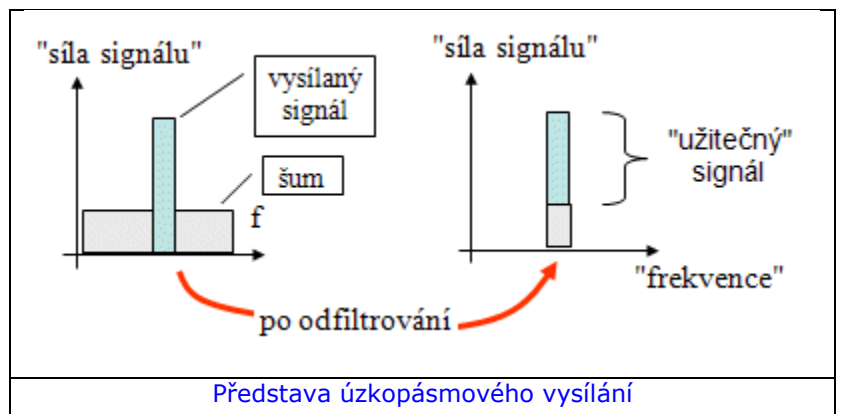
V roce 1985 takto americký regulátor uvolnil pro bezlicenční využití celkem tři rozsahy frekvencí ve dvou pásmech, označovaných jako „ISM“ (Industry, Science and Medical, tedy pro využití v oblasti průmyslu, vědy a zdravotnictví), a UNII (Unlicensed National Information Infrastructure).



Uvolněním popisovaných pásem pro bezlicenční využití poskytl americký regulátor mocný impuls všem snahám o bezdrátová řešení. Současně ale jednou částí svého rozhodnutí dost podstatně ovlivnil i jejich konkrétní podobu. Dal si totiž podmínku, že v těchto nově bezlicenčních pásmech mají být používána řešení „širokopásmová“, fungující na principu **rozpro-**

**stření do širšího spektra** (spread spectrum), která jsou podstatně šetrnější k životnímu prostředí.

Při klasickém vysílání (tzv. „úzkopásmovém“) je energie vyzařovaná vysílačem soustředěna do úzkého rozsahu frekvencí, a svou intenzitou musí převýšit úroveň šumu, neboť příjemce vyhodnocuje fakticky jen to, co přesáhne hladinu šumu.



Ovšem při "širokopásmovém" přenosu se využívají takové techniky, které umožňují, aby vysílač vysílal v širším rozsahu frekvencí (aby své vysílání doslova „rozprostřel do širšího spektra“), a mohl vysílat jen s nižší intenzitou, která ani nemusí přesahovat hladinu šumu. Podstatné je to, aby příjemce věděl, jakým způsobem k „rozprostření do šířky“ došlo, a uměl zpětně extrahovat užitečný signál, smíchaný s šumem. Naopak tomu, kdo potřebné informace o způsobu rozprostření nemá, se přijímaný signál jeví jen jako neužitečný šum. Výsledným efektem je pak větší robustnost přenosů, a naopak menší šance vzájemného ovlivňování různých přenosů. Ostatně, jednou z motivací amerického regulátora pro takovýto požadavek byla snaha co nejvíce omezit rušivé vlivy od jiných „zářičů“ ve stejných frekvenčních pásmech. Tedy například od mikrovlnných trub.

### Bezdrátový Ethernet

První snahy přenést do té doby drátový Ethernet i do bezdrátové podoby, a využít k tomu nově uvolněná frekvenční pásma, na sebe nenechaly dlouho čekat. Iniciátorem těchto snah byla společnost NCR, která měla poměrně zajímavou motivaci: vyráběla totiž různá zařízení, mezi nimi i pokladny (do obchodů), a hledala možnost jak je propojit mezi sebou, bez nutnosti „tahat dráty“. A tak z její iniciativy vzniká v roce 1988, na půdě společnosti IEEE (Institute of Electrical and Electronics Engineers) další pracovní skupina, aby se navrhovanému řešení věnovala. Jde tedy o stejnou platformu (IEEE), na které vznikaly a nadále vznikají standardy Ethernetu, v rámci pracovní skupiny IEEE 802.3. Ovšem bezdrátová varianta Ethernetu od začátku měla svou vlastní pracovní skupinu, očíslovanou jako IEEE 802.11.

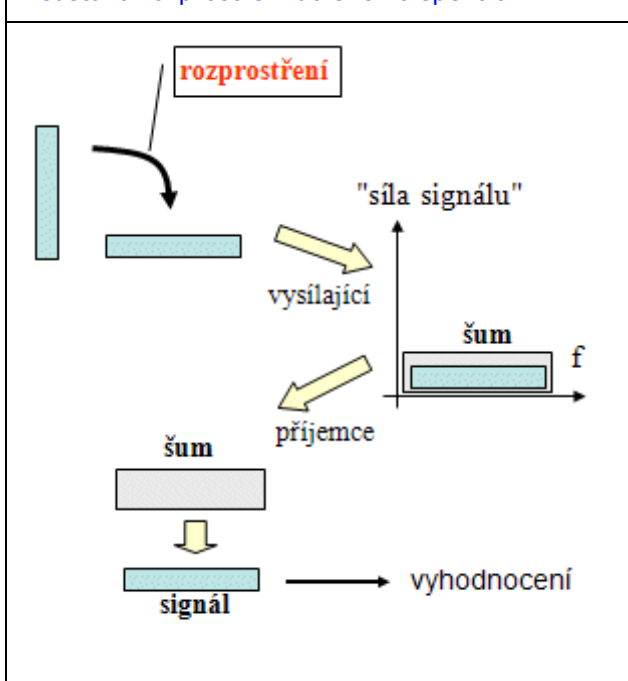
Teprve v roce 1997 se totiž podařilo najít v rámci pracovní skupiny 802.11 shodu na tom, jak by celé řešení mělo vypadat. Takže teprve tehdy se zrodil první standard bezdrátové varianty Ethernetu, označovaný podle svého původu jako „IEEE 802.11“. Nepleťme si ho ještě s dnešním Wi-Fi, protože do něj měl stále poměrně daleko. Byl to jen jeho předchůdce. Navíc mnohem méně výkonný. Dokázal totiž přenášet data (nominální) rychlostí jen 1 či 2 megabity za sekundu. Už měl ale vyvinuty ty přístupové metody, které používá i dnešní Wi-Fi. A pokud jde o možné techniky vysílání, alias řešení své fyzické vrstvy, zde tento první bezdrátový Ethernet nabízel dokonce více možností než dnešní Wi-Fi.

Ovšem už v době svého vydání byl tento standard bezdrátového Ethernetu zastaralý, a poměrně brzy byl nahrazen dalšími standardy. V roce 1999 tak spatřily světlo světa další dvě verze, a to IEEE 802.11a a IEEE 802.11b. No a ty už jsou tím, co dnes známe a používáme jako Wi-Fi.

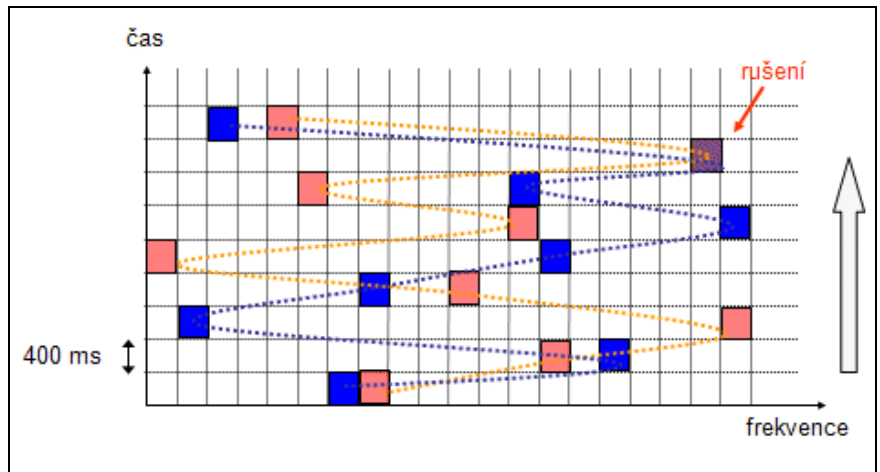
### Tři varianty řešení fyzické vrstvy

První varianta bezdrátového Ethernetu, kterou je standard IEEE 802.11 (pozor, ještě bez dalšího rozlišujícího písmene na konci), počítala celkem se třemi vzájemně alternativními možnostmi fungování své fyzické vrstvy, resp. způsoby vysílání a přenosu dat bezdrátovým éterem. Tedy vlastně se třemi různými technikami rozprostření do šířky (do širšího frekvenčního spektra).

Představa rozprostření do širšího spektra

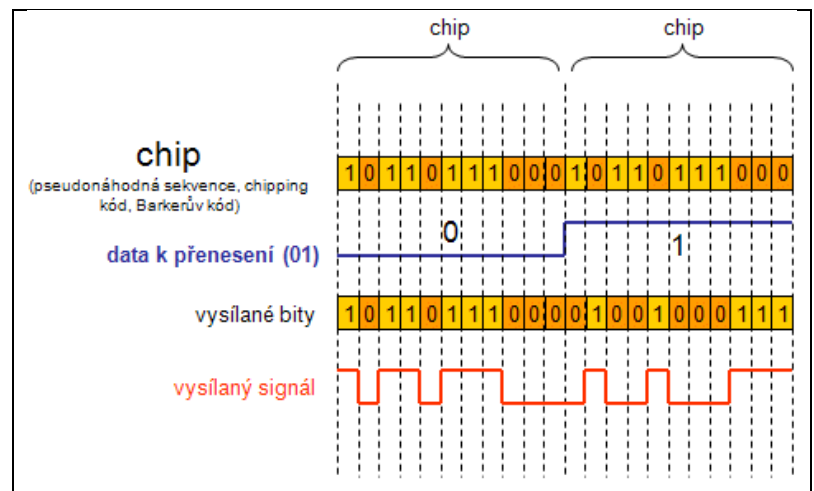


1. Jedna z těchto technik je označována jako tzv. **frequency hopping** (FHSS, Frequency Hopping Spread Spektrum). Jde o rychlé přeskakování z jedné frekvence na druhou, resp. z jednoho úzkého frekvenčního kanálu na jiný, způsobem který může vypadat jako zcela náhodný, ale oběma komunikujícím stranám je dopředu znám (je tzv. pseudonáhodný). Vysílač i přijímač (modrá) pak vždy „zůstávají“



- na každém jednotlivém frekvenčním kanále jen velmi krátkou dobu, načež oba současně přechází na jiný frekvenční kanál – a tak stále dokola. Výhoda je i v tom, že pokud mezi stejnými frekvenčními kanály „přeskakuje“ nějaký jiný přenos (červená), podle své vlastní pseudonáhodné posloupnosti kanálů, je šance že se oba přenosy ve stejném čase sejdou na stejných frekvencích relativně malá. A pokud k ní dojde, dají se dopady krátkého vzájemného zarušení snadno napravit. Nebo se dají i ignorovat, například u multimediálních přenosů, které si tolik nepotrpí na spolehlivost přenosů. Celou představu ukazuje následující obrázek. U bezdrátového Ethernetu (tj. u standardu IEEE 802.11) se předpokládalo poměrně pomalé přeskakování (hopping), a to 2,5krát za sekundu. Pro srovnání: stejná technika se dnes používá u technologie Bluetooth, ale zde dochází k přeskokům podstatně rychleji, 1600x za sekundu.
2. Další možností, kterou mohl bezdrátový Ethernet využívat, byly **infračervené přenosy** (DFIr, Difuse InfraRed), v pásmu 300 až 428 GHz. Ovšem tato varianta se používala zcela minimálně.
  3. Naopak jako nejperspektivnější se ukázala třetí varianta, označovaná jako „**přímé rozprostření do spektra**“, resp. „**s přímou modulací**“

**kódovou posloupností** (zkratkou DSSS, z anglického **DSSs, Direct Sequence Spread Spectrum**). Ta už nepočítá s přeskoky či jinými změnami frekvencí, ale s tím, že se trvale vysílá v širším rozsahu frekvencí. Navíc se **zde nevysílají přímo jednotlivé datové bity, ale místo nich celé sekvence bitů, označované jako tzv. chipy** (česky snad: úlomky). Můžeme si to představit tak, že pokud je třeba přenést jedničkový datový bit, vyšle se místo něj skupina bitů tvořících chip. Pokud se naopak má přenést jedničkový datový bit, vyšle se místo něj stejná sekvence bitů (chip), ale invertovaná.



### Řízení přístupu u bezdrátového Ethernetu

Stejně jako jeho „drátový“, musel se i bezdrátový Ethernet nějak vyrovnat se sdílenou povahou použitého přenosového média. Sdílenou proto, že podobně jako na původním koaxiálním kabelovém segmentu může i bezdrátově vysílat vždy jen jeden uzel (a ostatní mohou poslouchat). Ale jakmile začne vysílat více uzlů sou-

časně (ve vzájemném dosahu), dochází ke kolizi: jejich vysílání se „střetávají“ a vzájemně nenávratně promíchávají. U drátového Ethernetu řešila problém přístupu ke sdílenému média přístupová metoda CSMA/CD.

Předpokládala, že je možný příposlech nosné (CS, Carrier Sense), a tedy že každý uzel si může nejprve poslechnout, zda nevysílá někdo jiný.

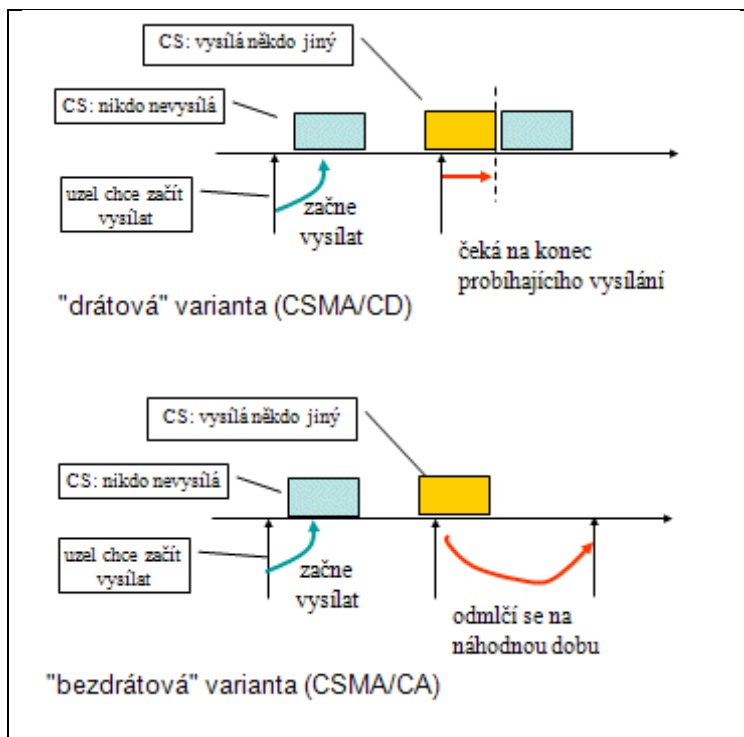
**U bezdrátového Ethernetu je příposlech nosné stále možný, a tak ho zájemci o vysílání používají.**

Jen se liší v tom, jak zareagují v případě, kdy zjistí že někdo již vysílá. U drátového Ethernetu, v rámci přístupové metody CSMA/CD, uzel čeká na konec vysílání, a jakmile toto skončí, začne vysílat sám. V případě bezdrátového Ethernetu je to naopak: **uzel, který zjistí že právě vysílá někdo jiný, to okamžitě „vzdá“, odmlčí se na náhodně dlouhou dobu** (takovou, jakou mu určil generátor náhodných čísel), **a teprve pak vše zkouší znovu, počínaje příposlechem nosné.** Rozdíl mezi oběma přístupy ukazuje obrázek.

Důvodem pro tuto zajímavou odlišnost je to, že **u bezdrátového vysílání nelze (jednoduše) detekovat kolize.** Rádiová rozhraní jsou totiž obvykle jen poloduplexní, v tom smyslu že buďto

přijímají nebo naopak vysílají, ale nikoli současně. Proto nedokáží vysílat a současně s tím skrze příjem monitorovat, zda v éteru nedošlo ke kolizi. Ostatně, proto také přístupová metoda bezdrátového Ethernetu není (a nemůže být) metodou CSMA/CD, neboli s detekcí kolize (CD, Collision Detect). Místo toho **se označuje jako CSMA/CA**, přičemž ono „CA“ na konci značí „Collision Avoidance“, neboli předcházení kolizím. Samotný příposlech nosné (CS, Carrier Sense) před začátkem vlastního vysílání sice dokáže počet kolizí poměrně významně zmenšit, ale ne zcela vyloučit.

**Jak se tedy v bezdrátovém Ethernetu řeší kolize, když k nim stále může docházet. Vše se řeší skrze potvrzování. Příjemce, který v pořádku přijal přenášená data, má povinnost poslat odesílateli potvrzení o jejich doručení. Pokud odesílatel nedostane takovéto potvrzení do určitého časového limitu, považuje to za ztrátu původně odeslaných dat (ať již kvůli kolizi či z jiného důvodu) a snaží se je odeslat znovu.**

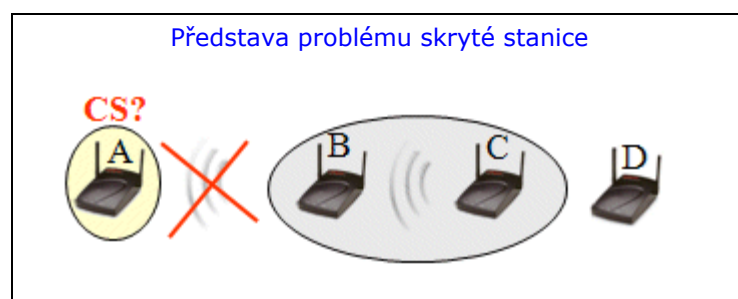


### Problém skryté a předsunuté stanice

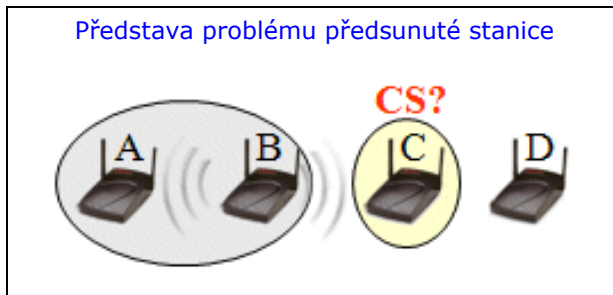
Řízení přístupu ke společně sdílenému médiu (éteru) má u bezdrátového Ethernetu (a obecně u bezdrátových přenosů jako takových) dva specifické problémy, obvykle označované jako problém skryté a předsunuté stanice. Lze si je nejlépe představit na následujících dvou obrázcích, se čtyřmi stanicemi A až D.

Na prvním obrázku vidíme **problém skryté**

**stanice:** stanice C zde vysílá, a stanice B její vysílání přijímá. Vysílání stanice C ale není tak silné, aby dosáhlo až ke stanici A. Pokud by ta chtěla začít vysílat, nejprve by pomocí příposlechu nosné (CS) dospěla k závěru, že právě nikdo nevysílá. A tak by stanice A začala vysílat sama, čímž by ale s velkou pravděpodobností narušila právě probíhající komunikaci mezi stanicemi B a C. Ty by to následně poznaly až z absence potvrzení správně přenesených dat.



Na druhém obrázku vidíme jinou situaci, která představuje **problém** tzv. **předsunuté stanice**. Zde stanice B komunikuje se stanicí A, ale její vysílání je všesměrové, a tak je může přijímat také stanice C. Ta, by sama, chtěla komunikovat se stanicí D, a to by mohla (vzájemná vysílání mezi B a A, a mezi C a D by se nerušila). Jenže když stanice C použije příposlech nosné (CS), zjistí, že někdo právě vysílá a tak sama vysílat nezačne, a podle pravidel přístupové metody CSMA/CA se odmlčí na náhodnou dobu. A teprve pak zkouší vše znovu.



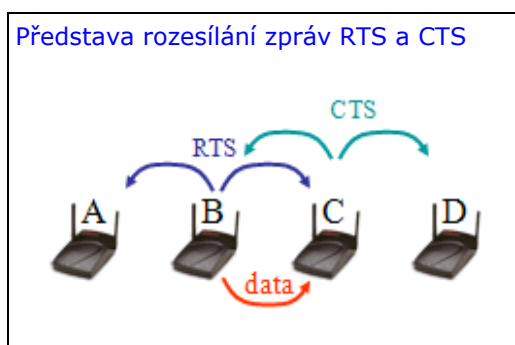
Až dosud popisované chování, na základě přístupové metody **CSMA/CA a bez ošetření hrozícího problému skryté a předsunuté stanice**, představuje nejjednodušší variantu ří-

zení přístupu u bezdrátového Ethernetu. Tato varianta je označována také jako metoda DCF, což je zkratka od Distributed Coordination Function.

**Pro řešení problému skryté a předsunuté stanice byla metoda DCF rozšířena o další variantu, označovanou jako „DCF se zprávami RTS/CTS“.** Její princip si lze představit tak, že když dvě stanice chtějí vzájemně komunikovat a předávat si data, nejprve upozorní své okolí (resp. všechny stanice, které se právě nachází v dosahu jejich vysílání), že tak budou činit a že je ostatní nemají vyrušovat. Navíc jim řeknou, jak dlouho bude jejich komunikace probíhat, a tak ostatní uzly v dosahu vědí, jak dlouho mají být zticha.

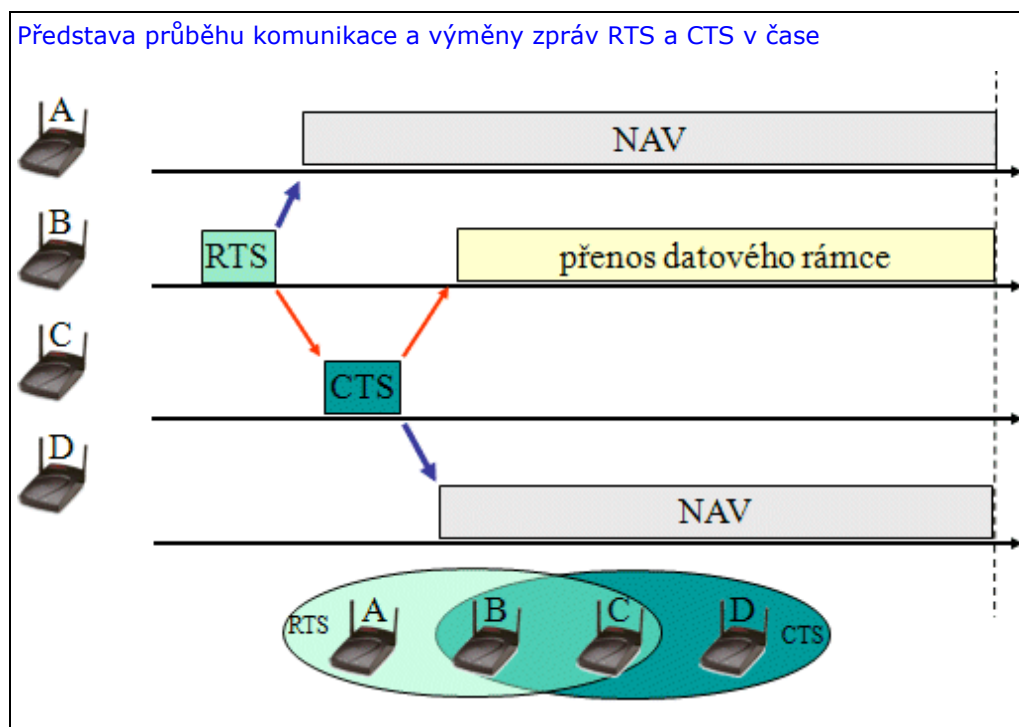
Konkrétní fungování této metody naznačují následující dva obrázky.

První ukazuje, jak stanice, která je iniciátorem komunikace, rozesle do svého okolí zprávu, v podobě **speciálního rámce RTS (Request To Send)**. Ten obsahuje vektor NAV (Network Allocation Vector), který není ničím jiným než údajem o tom, jak dlouho bude budoucí komunikace probíhat. Všechny stanice, které tento rámec přijmou, tedy na tuto dobu přestanou vysílat.



Zpráva v rámci RTS kromě vektoru NAV také identifikuje stanici C jako „tu, se kterou chce stanice B komunikovat“. A tak **stanice C**

**zareaguje vysláním „potvrzující“ zprávy CTS (Clear To Send), která také obsahuje (aktualizovaný) údaj o délce budoucího vysílání.** Tuto zprávu zachytí všechny stanice v dosahu stanice C, a díky ní vědí, že musí zůstat zticha (i na jak dlouho). Pak už komunikace mezi stanicemi B a C může probíhat, a ostatní stanice by do ní neměly zasahovat.

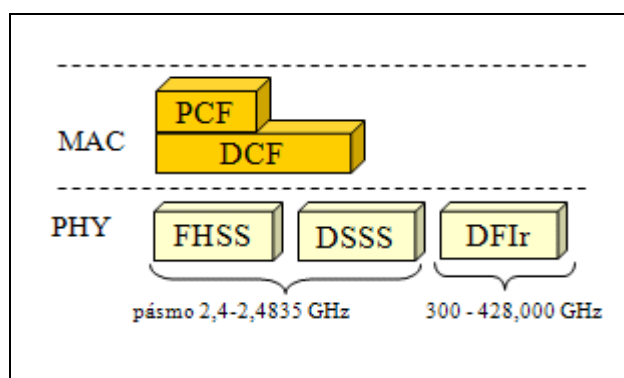




Ani uvedený postup se zprávami RTS a CTS však není schopen zaručit, že žádná jiná stanice nenaruší komunikaci, probíhající mezi uzly B a C (v našem případě). Již jen proto, že u bezdrátových přenosů není poloha stanic fixována, a tak se do blízkosti komunikujících stanic může dostat nějaká jiná stanice, která se neúčastnila původního dialogu se zprávami RTS a CTS a tudíž neví, že by měla být zticha.

### Další vývoj bezdrátového Ethernetu

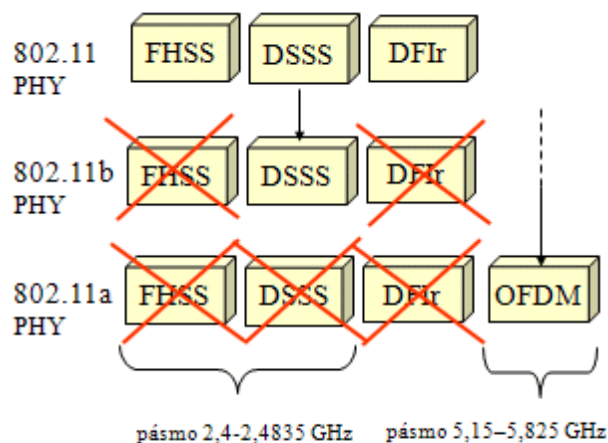
Zpět ale ke standardu IEEE 802.11, o kterém se ještě hovořilo jako o Ethernetu: následující obrázek shrnuje to, co jsme si o něm již řekli. Konkrétně že na úrovni fyzické vrstvy nabízí **tři alternativní řešení**, představující tři různé varianty bezdrátových přenosů: prostřednictvím techniky infračerveného přenosu, pomocí frekvenčních přeskoků (Frequency Hopping, FHSS), a s přímým rozptřením do širšího spektra (Direct Sequence Spread Spectrum, DSSS). Nezapomínejme na to, že rychlostní maximum všech těchto variant byly 2 Mbit/s za sekundu.



Nad trojím možným řešením fyzické vrstvy pak byla implementována distribuovaná metoda řízení přístupu DCF (Distributed Coordination Function), fungující na principu CSMA/CA. Případně ještě metoda PCF (Point Coordination Function), která je centralizovaná a řízená: předpokládá, že jedna konkrétní stanice je pověřena řízením, a že ostatním stanicím přiděluje právo vysílat na základě jejich žádostí. Ale tato metoda přístupu je i dnes velmi vzácná.

Standard IEEE 802.11, přijatý v roce 1987 a označovaný jako bezdrátový Ethernet, byl zastaralý už v době svého přijetí. Snahy o jeho zdokonalení se ubíraly především směrem ke zrychlení datových přenosů, a byly úspěšné. Vyžadovaly ovšem poměrně zásadní změnu ve způsobu realizace fyzické vrstvy, neboť právě ona rozhoduje o rychlosti přenosu dat.

Proto když v roce 1999 spatřily světlo světa **dva další standardy, IEEE 802.11a a IEEE 802.11b, řešení jejich fyzické vrstvy bylo podstatně jiné**. Jak se lišilo, naznačuje následující obrázek:

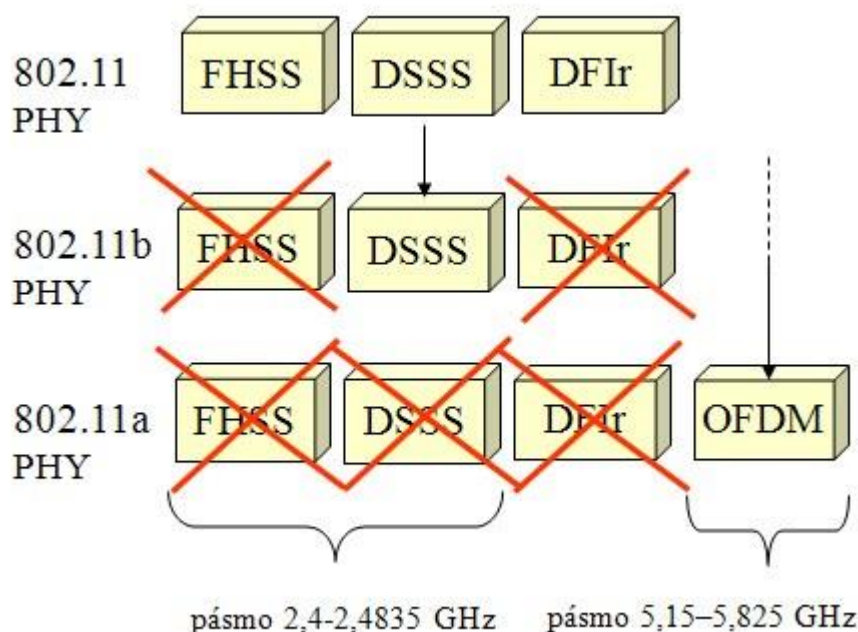


V ČR se mohou používat produkty, vycházející z obou těchto variant, i když u druhé z nich jen s určitými omezeními. Už se ale v souvislosti s nimi nehovoří o bezdrátovém Ethernetu, protože tak se označoval jen původní standard IEEE 802.11 z roku 1997 (s max. rychlostí 2 Mbit/s). Spíše se o nich mluví jako o Wi-Fi, i když ani toto označení není vůbec přesné. Samotné Wi-Fi je totiž něčím jako nálepkou, stvrzující že konkrétní produkt vyhovuje standardu IEEE 802.11a nebo 802.11b a je plně kompatibilní a interoperabilní s jinými produkty, které vychází ze stejných standardů.

## IEEE 802.11b a IEEE 802.11a

Ovšem už v době vydání prvního standardu bezdrátového Ethernetu bylo jeho řešení vnímáno jako překonané, a intenzivně se pracovalo na řešení novém a ještě rychlejším. To se podařilo najít a standardizovat poměrně rychle, již v roce 1999. Jak jsme si již také uvedli v předchozím dílu tohoto seriálu, jednalo se o standardy IEEE 802.11a a IEEE 802.11b, které skutečně dosáhly poměrně výrazného zrychlení, ovšem za cenu poměrně významné změny rádiové části celého řešení. Ukazuje to i následující obrázek, který jsme si ukazovali i minule, a podle kterého varianta IEEE 802.11b využívá ze tří původních technik bezdrátového rádiového přenosu již jen jednu (techniku DSSS, Direct Sequence Spread Spektrum), zatímco varianta IEEE 802.11a dokonce přišla s další technikou, tzv. **ortogonálního frekvenčního multiplexu (OFDM)**.

Představa řešení fyzické vrstvy u IEEE 802.11



Podstatné také je, že varianta **IEEE 802.11b**, dodnes asi stále nejrozšířenější a nejpoužívanější po celém světě, nadále **funguje v bezlicenční pásmu 2,4 GHz**, kde dosahuje **nominální přenosové rychlosti až 11 Mbitů**. Naproti tomu varianta **IEEE 802.11a** se již posunula do **pásmu 5 GHz**, kde dokáže pracovat **rychlostí až 54 Mbit/s** (opět ale jen **nominálně**).

## Nominální vs. efektivní přenosová rychlost

Jen pro připomenutí, **nominální rychlost** vypovídá spíše o tom, **jak dlouho trvá přenos jednoho bitu, bez ohledu na to, zda jde o bit užitečný či režijní**. Tzv. **efektivní přenosová rychlost**, která bere v úvahu již jen **užitečná data**, bývá v praxi menší, a to až o desítky procent, protože už bere v úvahu nejrůznější režii a také všelijaké nutné prodlevy, opakované přenosy apod. Může být a bývá závislá i na konstrukci rádiových zařízení, což naznačují nejrůznější praktické testy a srovnání, které sledují chování různých produktů ve stejných podmínkách. Obvykle se jimi dosahované rychlosti liší i dosti podstatně.

Nečekejme tedy od technologie, která deklaruje (nominální) přenosovou rychlost 11Mbitů za sekundu, že třeba za minutu přenese  $60 \cdot 11$  megabitů. Bude to podstatně méně, a rozdíl jde právě na úkor rozdílu mezi nominální (a tedy spíše teoretickou) rychlostí a rychlostí efektivní (skutečně dosahovanou).

Následující tabulka uvádí příklad dosahovaných efektivních rychlostí pro technologie na bázi IEEE 802.11b, 802.11a a 802.11g (viz dále). Samozřejmě nejde o žádné závazné a oficiální hodnoty, ale spíše o orientační a zprůměrované výsledky různých praktických testů.

Standard	Max. nominální rychlost	Reálná efektivní rychlost
802.11b	11 Mbit/s	do 6 Mbit/s
802.11g	54 Mbit/s	do 22 Mbit/s
802.11a	54 Mbit/s	do 25 Mbit/s

### Není rychlost jako rychlost

Úvaha o rozdílu mezi nominální a efektivní rychlostí přitom platí obecně, pro všechny technologie, drátové i bezdrátové. Ovšem bezdrátové technologie se navíc musí potýkat i s tím, že se jim podmínky pro přenos mohou měnit doslova „pod rukama“ a velmi dynamicky. Například v důsledku jejich pohybu, aktivit jiných stanic či jen v důsledku nejrůznějšího rušení, často předem neodhadnutelného a nepredikovatelného.

Řada bezdrátových technologií proto dokáže pracovat i nižšími (nominálními) rychlostmi, což je činí přeci jen odolnějšími vůči různým nepříznivým vlivům. Navíc se nominální rychlosti mohou měnit poměrně dynamicky, na základě vlastního uvážení bezdrátových stanic, které průběžně monitorují podmínky pro přenos. Pokud zjistí jejich významnější zhoršení (či zlepšení), mohou samy a ihned přejít na nižší (či vyšší) nominální přenosovou rychlost.

První technologie bezdrátového Ethernetu, ještě podle standardu IEEE 802.11 (z roku 1997 a bez rozlišujícího písmene), dokázala pracovat jen dvěma různými nominálními rychlostmi, a to 1 a 2 Mbit/s. Standard IEEE 802.11b z roku 1999 již počítal s širším výběrem možných nominálních rychlostí: 1, 2, 5.5 a 11 Mbit/s. Stejně starý standard IEEE 802.11a, který předpokládá fungování v pásmu 5 GHz, počítá s ještě větší škálou možných (nominálních) přenosových rychlostí: 6, 9, 12, 18, 24, 36, 48 a 54 Mbit/s.

### Frekvenční kanály v bezlicenčních pásmech

Dosahování co možná nejvyšších (nominálních i efektivních) přenosových rychlostí je u technologií IEEE 802.11 komplikováno také tím, že rozsahy frekvencí v příslušných bezlicenčních pásmech jsou omezené, a to poměrně výrazně. Například když se hovoří o tom, že IEEE 802.11b pracuje „v pásmu 2,4 GHz“, ve skutečnosti to znamená jen možnost fungování v rozsahu 2,400 až 2,4835 GHz, neboli v pásmu širokém pouze 83,5 MHz. A toto pásmo je navíc v různých částech světa různé, viz dále.

Podobně „v pásmu 5 GHz“ znamená (např. v ČR) ve třech různých pásmech:

1. 5,150 až 5,250 GHz (šířka 100 MHz)
2. 5,250 až 5,350 GHz (šířka 100 MHz)
3. 5,47 až 5,725 GHz (šířka 255 MHz)

Přitom technologie IEEE 802.11b potřebují ke své práci frekvenční kanály o šířce 22 MHz. Na otázku, kolik se jich „vejde“ do pásma 2,400 až 2,4835 GHz, alias 83,5 MHz, je jednoduchá odpověď: pokud se nemají překrývat, pak jen tři! To znamená, že v celém tomto bezlicenčním pásmu, vyhrazeném pro bezdrátové přenosy, jsou jen tři nepřekrývající se frekvenční kanály, a tudíž prostor jen pro tři vzájemně se neovlivňující (nerušící) přenosy!

Praxe ale počítá s přeci jen větším počtem kanálů, které ovšem „nahušťuje“ do stejné široké oblasti – takže se přitom nutně z části překrývají. Konkrétně tak, že odstup mezi těmito kanály, širokými 22 MHz, je pouze 5 MHz. Příslušné přenosy na takto „sousedních“ frekvenčních kanálech pak jsou stále možné, ale vzhledem k částečnému překrytí nedosahují takových výsledků, jako při využití nepřekrývajících se kanálů.

Kolik takovýchto vzájemně se překrývajících kanálů je k dispozici, je závislé na šířce celého bezlicenčního „pásmu 2,4 GHz“, vyhrazeného pro tyto účely. A jak jsme si už řekli, v různých částech světa je situace různá. Jak moc, ukazuje následující tabulka:

Kanál č.	Rozsah frekvencí	USA	Evropa	Japonsko
1	2401-2423	x	x	x
2	2406-2428	x	x	x
3	2411-2433	x	x	x
4	2416-2438	x	x	x
5	2421-2443	x	x	x
6	2426-2448	x	x	x
7	2431-2453	x	x	x
8	2436-2458	x	x	x
9	2441-2463	x	x	x
10	2446-2468	x	x	x
11	2451-2473	x	x	x
12	2456-2478	-	x	x
13	2461-2483	-	x	x
14	2466-2488	-	-	x

**V ČR**, podle posledního rozhodnutí (všeobecného oprávnění VO-R/12/08.2005-6) našeho správce frekvenčního spektra (Českého telekomunikačního úřadu) **máme k dispozici celkem 13** takovýchto **frekvenčních kanálů**. Z nich pak můžeme vybrat 3 vzájemně se nepřekrývající kanály, a to 1, 7 a 13, které navíc mají i poněkud větší odstupy mezi sebou. Obdobně je tomu i ve většině zemí Evropy.

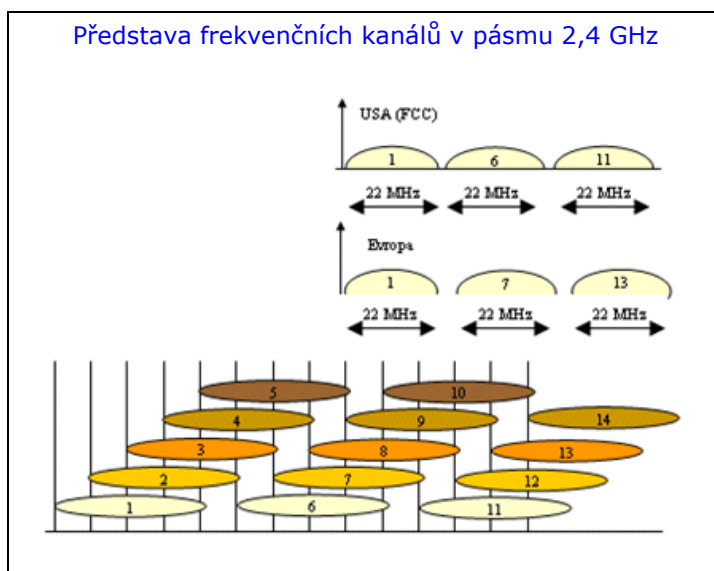
Naopak v USA mají k dispozici jen poněkud užší celkový rozsah frekvencí, do kterého se jim „vejde“ pouze 11 vzájemně se překrývajících kanálů o šířce 22 MHz. Tři nepřekrývající se z nich stále lze vybrat, ale jsou jiné než v Evropě: jsou to kanály 1, 6 a 11. Jejich odstupy mezi sebou jsou menší než u „evropské trojice“, a tak je zde přeci jen větší nebezpečí vzájemného ovlivňování jednotlivých přenosů.

Situace „pásmu 5 GHz“ je přeci jen příznivější. Jak jsme si již uvedli výše, zde jsou k dispozici fakticky tři rozsahy frekvencí, o 100 + 100 + 255 MHz. Navíc technologie IEEE 802.11, které v těchto pásmech pracují, používají poněkud užší frekvenční kanály, a to o šířce 20 MHz. Takže i kanálů, které se nepřekrývají, zde mají k dispozici přeci jen více.

#### **Bezlicenční neznamená bez pravidel**

Připomeňme si nyní ještě jednu zásadní skutečnost, se kterou jsme se seznámili již v předchozích dílech tohoto seriálu. Jde o samotný bezlicenční princip využití určitých frekvencí.

**Tento princip říká, že ten, kdo chce tyto frekvence využít, k tomu nepotřebuje žádné individuální oprávnění (dříve: individuální licenci). Místo toho stačí, když správce kmitočtového spektra vydá jedno „společné“ rozhodnutí, u nás ve formě tzv. všeobecného oprávnění (dříve: generální licence), a v něm stanoví podmínky, za**



kterým může kdokoli předmětné frekvence využít. Rozhodně to tedy neznamená, že by si na frekvencích, určených pro takovéto bezlicenční využití mohl kdokoli dělat cokoli uzná za vhodné. To opravdu nemůže.

Podmínky, které stanovuje všeobecné oprávnění (generální licence), mohou být i dosti přísné, a jejich nedodržení může být sankcionováno vysokými pokutami. Takže pozor na to, a vždy se nejprve seznamte s jejich zněním. U zde popisovaných pásem a technologií je relevantní rozhodnutí regulátora (ČTÚ) v podobě již zmiňovaného všeobecného oprávnění VO-R/12/08.2005-6, které lze nalézt (například) na webu ČTÚ.

**Nejvýznamnější podmínkou tohoto (i obdobných všeobecných oprávnění) je omezení toho, s jakou „sílu“ (tzv. vyzářeným výkonem) mohou vysílače vysílat v tom kterém frekvenčním pásmu. Tak například v „pásmu 2,4 GHz“, přesněji v rozsahu 2,400 až 2,4835 GHz, nesmí být překročeno 100 mW EIRP (Effective Isotropical Radiated Power, alias „efektivní izotropický vyzářený výkon“).**

**V případě „pásmu 5GHz“** je to ještě složitější, protože zdejší tři dílčí pásma se liší právě tím, jak silné je u nich omezení vysílacího výkonu:

**pásmo 5,150 až 5,250 GHz: možnost využití pouze uvnitř budov, max. 200 mW EIRP**

**pásmo 5,250 – 5,350 GHz: možnost využití pouze uvnitř budov, max. 200 mW EIRP s regulací výkonu, jinak o 3 dB méně**

**pásmo 5,47 až 5,725 GHz: možnost využití i vně budov, max. 1 W EIRP s regulací výkonu a výběrem frekvencí (DFS a TPC), jinak o 3 dB méně**

### Omezení výkonu

Podrobnější vysvětlení toho, co vlastně je EIRP (efektivní izotropický vyzářený výkon) by asi vydalo na samostatný díl tohoto seriálu, proto jen velmi stručně a s velkým zjednodušením: jde o výkon, vyzářený do prostoru ideální bodovou anténou (která vyzařuje do všech směrů a stejně). Takováto ideální bodová anténa ale v praxi neexistuje, protože reálné antény vždy vyzařují v některých směrech více a v jiných méně. Tam, kde reálné antény vyzařují v určitém směru více, než ideální bodová anténa, se hovoří o jejich zisku. Čím větší má anténa zisk, tím intenzivněji vyzařuje v příslušném směru – ale toto její vyzařování nesmí překročit mezní hodnotu, stanovenou ve všeobecném oprávnění pro bezlicenční pásmo.

Výpočet:

Na zařízení je výkon uveden v jednotkách dBm. Vzorec pro převod:  $\text{mW} = 10^{\text{dBm}/10}$ .

Na zařízení je výkon uveden v jednotkách mW. Vzorec pro převod:  $\text{dBm} = 10 \cdot \log_{10}(\text{mW})$

Konkrétní „sílu“ vyzařování v praxi vypočítáme tak, že sečteme výstupní výkon vysílače (neboli „sílu“ toho, co vystupuje z vysílače), od toho odečteme ztráty vzniklé na kabelovém vedení k anténě, a naopak přičteme zisk použité antény. Výsledkem bude číslo (v dBm), které se přepočítá na mW a porovná s maximální povolenou hodnotou.

**V praxi** je třeba dávat pozor hlavně na situace, kdy se chystáte použít externí anténu s velkým ziskem (typicky směrovou anténu), protože zde nejspíše bude třeba snížit výstupní výkon vysílače. Pokud si kupujete nějaké „krabicové“ zařízení s integrovanou anténou, pak nejspíše nemusíte mít žádný strach z překročení stanovených limitů.

### IEEE 802.11g a IEEE 802.11h (oba z roku 2003).

První z nich, IEEE 802.11g, si můžeme představit jako vylepšenou verzi 802.11b: stále funguje v pásmu 2,4 GHz, na stejných frekvenčních kanálech jako 802.11b, ale díky dokonalejšímu a vyspělejšímu technologickému řešení (mimo jiné i díky technice OFDM) zde dokáže dosahovat maximální (a samozřejmě jen nominální) rychlosti 54 Mbit/s. V případě horších podmínek pro přenos pak může fungovat i s nižšími nominálními rychlostmi. Konkrétně 1, 2, 5.5, 6, 9, 11, 12, 18, 22, 24, 33, 36 a 48 Mbit/s.

Druhý je standard **IEEE 802.11h**. Ten navazuje na předchozí standard **IEEE 802.11a**, a pracuje obdobně – v pásmu **5 GHz** a s maximální (nominální) rychlostí 54 Mbit/s. Od svého předchůdce se liší v tom, že povinně podporuje dva mechanismy, napomáhající k „ekologičtějšímu“ chování v éteru. Jde konkrétně o **schopnost dynamické volby frekvencí (DFS, Dynamic Frequency Selection)**, která vybírá takové frekvenční kanály, které jsou volné, resp. kde neprobíhá jiný provoz. Dále jde o schopnost regulovat vysílací výkon (**TPC, Transmit Power Control**), tak aby tento přesně odpovídal potřebám požadované komunikace a nebyl zbytečně vyšší (což by zvyšovalo možnost rušení jiných přenosů).

Zmínka o nutnosti použít regulaci výkonu a výběr frekvencí se týkala právě volby mezi standardy IEEE 802.11a a IEEE 802.11h. V případě 802.11h, který má obě funkce, lze vysílat plným (výše uvedeným) výkonem, tj. 200 mW, resp. 1 W. Pokud ovšem zařízení těmito mechanismy vybaveno není (jako například zařízení dle standardu 802.11a), mohou vysílat jen s výkonem o 3 dB nižším, neboli polovičním (protože -3 dB znamená poměr 1:2).

### **Standard IEEE 802.11n a další vývoj.**

Vývoj standardů v rámci pracovní skupiny IEEE 802.11 samozřejmě neskončil verzemi „h“ a „g“, ale pokračuje dál. Je z jedné strany motivován neutuchající poptávkou po ještě rychlejších bezdrátových přenosech, a z druhé strany dostupností nových technik, které rychlejší přenosy umožňují.

Jednou takovou technikou, na kterou sází jak připravované verze standardů IEEE 802.11 (pro WLAN), tak třeba i **technologie WiMAX** (pro nasazení spíše v rozlehlejších sítích včetně sítí metropolitní), a také budoucí **mobilní technologie 4G**, je technika označovaná jako **MIMO**. Její kuriózní název je zkratkou od „Multiple Input and Multiple Output“, a zaslouží si malé zastavení a vysvětlení.

U klasických rádiových přenosů jsou různé odrazy (odražené vlny, například od zdí apod.) nežádoucí a zhoršují celkovou kvalitu příjmu. Nicméně dostatečně inteligentní a propracovaný přijímač dokáže takovéto odražené vlny naopak využít ke zlepšení příjmu, díky tomu že je dokáže zbavit jejich „chybovosti“ a jakoby je započítat k užitečnému signálu. Teorie kolem toho je velká a složitá, ale v praxi to funguje.

Co ale **tato technika ke svému fungování nutně potřebuje, je více antén**. Samotný **princip MIMO** tedy počítá s více anténami (například se dvěma či třemi) na straně vysílače i přijímače. Proto také ono „Multiple Out“, čímž se míní více antén na straně vysílače (a tedy „out“ znamená směrem ven do éteru), a „Multiple In“, čímž se míní naopak více antén na straně přijímače. Systémy MIMO tak v praxi poznáme poměrně snadno, podle většího počtu antén, či spíše malých antének. Mobily, které využívají techniku MIMO k dosažení rychlejších mobilních datových přenosů, je už nejspíše budou mít integrované ve svém těle, a nebudou z něj nijak vyčuhovat.

Ovšem dnes jsme s aplikací techniky MIMO stále ještě na začátku, a to i v oblasti sítí WLAN, vycházejících ze standardů od pracovní skupiny IEEE 802.11. Ta na novějším a ještě rychlejším standardu **IEEE 802.11n** pracuje již od roku 2004. Na trhu existují konkrétní produkty od konkrétních výrobců, kteří deklarují, že jejich zařízení již vychází ze standardu IEEE 802.11n a fungují s rychlostí 100 Mbit/s, či dokonce ještě rychleji.

### **Co je a co není Wi-Fi?**

Připravit specifikace určitého řešení a vydat je ve formě standardu je jen jedním z kroků, které jsou nutné pro prosazení určitého řešení do života a každodenní praxe. Sebelepší standard nebude nic platný tam, kde konkrétní implementace nebudou tento standard dostatečně přesně dodržovat. Nebo ho sice budou dodržovat, ale přesto si různé produkty od různých výrobců nebudou rozumět navzájem. A právě u bezdrátových řešení, kde se spolu velmi často setkávají a chtějí vzájemně komunikovat produkty od různých výrobců, je jejich **vzájemná kompatibilita (slučitelnost) a interoperabilita (schopnost vzájemné spolupráce) velmi důležitá**.





Zdůrazněme si ještě jednou to nejpodstatnější: „Wi-Fi“ nejsou všechny produkty, vycházející ze standardů IEEE 802.11, ale jen některé z nich.

Pravdou ale je, že v praxi se o všech takovýchto produktech hovoří jako o Wi-Fi, bez ohledu na to zda mají či nemají příslušnou certifikaci. Takže třeba když si kupujete nový notebook, zajímáte se, jestli má zabudované Wi-Fi. Přitom byste se měli zajímat hlavně o to, zda má zabudované rozhraní dle některého ze standardů IEEE 802.11 (a také kterého), a zda takovéto rozhraní má příslušnou Wi-Fi certifikaci. Teprve pak totiž máte dostatečnou záruku, že si bude dobře rozumět se zařízeními od jiných výrobců.



# Architektura (topologie) Wi-Fi sítí

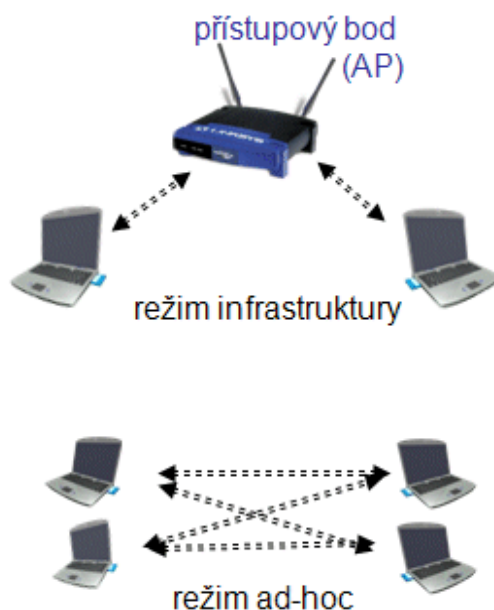
## Režim infrastruktury a režim ad-hoc

V případě bezdrátových sítí Wi-Fi máme v zásadě stejné dvě možnosti propojení, jako u drátového Ethernetu. Jen se jim pochopitelně říká jinak. Označují se totiž jako „režimy“, a to:

1. režim infrastruktury (infrastructure mode)
2. režim ad-hoc (ad-hoc mode)

V případě režimu infrastruktury je koncový uzel označován jednoduše jako „stanice“ (či: koncová stanice), zatímco analogií rozbočovače (HUB) je zde tzv. přístupový bod (v angličtině Access Point, zkratkou AP). Beze změny, oproti drátové variantě, je to že v tomto režimu veškerá komunikace vždy prochází přes přístupový bod, zatímco koncové stanice nikdy nekomunikují přímo mezi sebou.

Představa režimů infrastruktury a ad-hoc u Wi-Fi



Naopak v režimu ad-hoc žádný přístupový bod (AP) není, a jednotlivé koncové stanice zde komunikují mezi sebou přímo. Díky bezdrátovému charakteru může jít o komunikaci mezi více dvojicemi koncových uzlů (nikoli ale ve stejném čase). V případě drátového propojení by k něčemu takovému bylo zapotřebí u každého uzlu více ethernetových rozhraní, ale u bezdrátového řešení stačí jen jediné rádiové rozhraní. Proto si režim ad-hoc můžeme představit, v souladu s předchozím obrázkem, jako komunikaci mezi více koncovými stanicemi, ovšem vždy jen „po dvojicích“ a přímo, bez prostředníka.

## Autentizace a asociace u Wi-Fi

Drátové lokální sítě na bázi Ethernetu mají zajímavou přednost v tom, že u nich je už pouhým propojením (vedením kabelů) pevně a jednoznačně dána vazba mezi koncovými uzly a rozbočovači. Případně mezi dvěma koncovými uzly, u přímého propojení.

U bezdrátových sítí Wi-Fi je ale všechno jinak. Zde je vazba mezi koncovými stanicemi a přístupovým bodem podstatně volnější a také dynamická, protože se může měnit v čase. (stanice se mohou pohybovat). Jak ale tato vazba vlastně vzniká? Čím je nahrazena logická vazba mezi přístupovým bodem a koncovou stanicí, při absenci „drátu“, který by ji jednoznačně určil?

Přístupové body Wi-Fi jsou za tímto účelem vybaveny řadou služeb, mezi které patří (mimo jiné) schopnost: **autentizace**: v rámci této funkce přístupový bod zjišťuje, o jakou stanici jde, resp. zda je skutečně tím, za koho se vydává.

**asociace:** v rámci této funkce dochází ke vzniku logické vazby mezi přístupovým bodem a konkrétní stanicí. Stanice je „přidružena“ (tzv. asociována) k danému přístupovému bodu.

**de-asociace:** opak asociace, dochází k uvolnění (zrušení) vazby mezi přístupovým bodem a koncovou stanicí.

Službu asociace si lze s trochou fantazie představit jako analogii propojení rozbočovače a koncového uzlu pomocí vhodného kabelu, a službu deasociace naopak jako rozpojení takového kabelu. Jen místo fyzické manipulace s kabely jsou tyto funkce realizovány na logické úrovni.

Dalším rozdílem je pak to, že u fyzického propojení, pomocí kabelů, se o oprávněnosti provést takové propojení rozhoduje také „fyzicky“, a lze jej účinně a efektivně blokovat zamezením přístupu k příslušným technickým prostředkům (například uzamknutím příslušné rozvodové skříně apod.).

U bezdrátových sítí je to opět komplikovanější, neboť vše se musí řešit na logické úrovni. Proto se u Wi-Fi sítí setkáme s tím, že **přístupové body musí poměrně detailně a důsledně zjišťovat, co jsou zač stanice**, které se s nimi chtějí asociovat (spojit a komunikovat). Právě k tomu slouží již zmiňované služby **autentizace**. Ty jsou nejčastěji řešeny **dvěma základními způsoby**:

1. **otevřenou autentizací** (Open System Authentication): zde se fakticky nezkoumá, co je stanice zač.
2. **autentizací pomocí sdíleného klíče** (Shared Key Authentication): zde musí stanice prokázat, že vlastní požadovaný klíč (stejný, jaký vlastní přístupový bod, resp. jaký je na něm nastaven, proto „sdílený klíč“, anglicky: shared key).

## Zabezpečení sítě

### Autentizace pomocí WEP

**WEP (Wired Equivalent Privacy**, tj. *soukromí ekvivalentní drátovým sítím*) je v označení pro **zastaralé zabezpečení bezdrátových sítí podle původního standardu IEEE 802.11**. Cílem WEP bylo poskytnout **zabezpečení obdobné drátovým počítačovým sítím** (např. kroucená dvojlinka), protože rádiový signál je možné snadno odposlouchávat i na delší vzdálenost bez nutnosti fyzického kontaktu s počítačovou sítí. WEP byl prolomen v srpnu 2001.

**V bezdrátové počítačové síti pracuje WEP na linkové vrstvě, kde šifruje přenášené rámce pomocí proudové šifry RC4**. Díky tomu není možné snadno číst odposlechnutý datový provoz přenášený v bezdrátové síti a není nutné měnit stávající síťové aplikace ani přenosové protokoly, protože šifrování je z jejich hlediska transparentní. Takzvaný 64bitový WEP používá 40bitový klíč, ke kterému je připojen 24bitový inicializační vektor a dohromady tak tvoří 64bitový RC4 klíč. Delší 128bitový WEP používá 104bitový klíč, ke kterému je připojen 24bitový inicializační vektor a dohromady tak tvoří 128bitový RC4 klíč. Někteří výrobci bezdrátových zařízení poskytují obdobným způsobem i 256bitový WEP. Pro ověření správnosti dat používá WEP kontrolní součty CRC-32.

### WEPplus

WEPplus (někdy označován jako WEP+) je **vylepšení původního WEP** zabezpečení od Agere Systems, **které se snaží odstranit takzvané slabé inicializační vektory**, pomocí kterých může útočník velmi rychle spočítat použitý šifrovací klíč použité proudové šifry RC4 a může tak nejen bezdrátový provoz odposlouchávat, ale může se do bezdrátové sítě zabezpečené pomocí WEP i připojit. Pokud však není WEPplus na všech komunikujících stranách v bezdrátové síti, nemá toto zabezpečení výhody oproti běžnému WEP.

### WEP2

**Vylepšení původního WEP** zabezpečení, které se snaží odstranit bezpečnostní nedostatky původního WEP. WEP2 **rozšiřuje inicializační vektory a zesiluje 128bitové šifrování**. Používal se na zařízeních, na kterých nebylo možné provozovat novější **WPA** nebo WPA2 zabezpečení. WEP2 má však stejné bezpečnostní problémy jako WEP, jen útočníkovi zabere více času.

## Autentizace pomocí WPA

**WPA (Wi-Fi Protected Access, tj. chráněný přístup k Wi-Fi)** je obchodní označení pro zabezpečení bezdrátových sítí. Po prolomení zabezpečení WEP v roce 2001 definovala Wi-Fi Alliance v roce 2002 zabezpečení WPA pro Wi-Fi sítě jako část tehdy připravovaného standardu IEEE 802.11i. (Wi-Fi Alliance vlastní práva na značku Wi-Fi i WPA a certifikuje zařízení, která ji nesou.)

Pro eliminaci slabých míst zabezpečení byl vyvinut protokol **TKIP (Temporal Key Integrity Protocol)**, který dočasně odstranil problém s inicializačními vektory a zavedl dynamickou správu šifrovacích klíčů, které jsou pomocí něj mezi klientem a přístupovým bodem bezpečně přenášeny nejen na začátku komunikace, ale i během ní. Na straně klienta (počítače, který se připojuje k bezdrátové síti) je nasazen tzv. suplikant (žadatel), což je univerzální „démon“ běžící v pozadí na hlavním procesoru počítače a zajišťující autentizaci klienta a správu šifrovacích klíčů pomocí TKIP.

Pro starší klientská zařízení (tj. bezdrátové síťové karty v počítačích) bylo možné provést aktualizaci obsluhovaného software (typicky ovladač, firmware a obsluha bezdrátových zařízení v operačním systému spolu s výše zmíněným suplikantem) pro zařízení vyrobená po roce 1999. Na straně přístupových bodů (AP) byla možná aktualizace software pro WPA až pro zařízení vyráběná po roce 2003 (z důvodu větší náročnosti na jejich výpočetní výkon pro implementaci TKIP).

WPA používá 128bitový šifrovací klíč a 48bitový inicializační vektorem (označován zkratkou IV), takže i když používá stejnou šifru jako WEP, odolává lépe útokům, jimiž je WEP napadán.

WPA vylepšuje kontrolu integrity dat (pro snadnou možnost vyřazení poškozených rámců). WPA používá kontrolu pomocí **MAC (Message Authentication Code, konkrétně algoritmus nazvaný Michael)**, která je zde nazývána MIC (Message Integrity Code). MIC metoda použitá ve WPA zahrnuje počítadlo rámců, které chrání před útoky snažícími se zopakovat předchozí odposlouchanou komunikaci.

Autentizace klienta je pro WPA navržena buď pro použití „předsdílené fráze“ (tzv. PSK – *Pre-shared key*) nebo pro použití s autentizačním serverem (typicky RADIUS) pomocí protokolu IEEE 802.1X.

### Vady WPA

Navzdory prodloužení klíčů a inicializačních vektorů, snížení počtu zaslaných paketů s příbuznými klíči a systému ověřujícím integritu zpráv je dnes snadné WPA prolomit, pokud je použito společně s TKIP. Proto je použití této kombinace protokolů považováno za stejně nebezpečné jako u překonaného protokolu WEP a je doporučeno používat bezpečnější protokol jako WPA2 (CCMP a AES).

WPA nefunguje v tzv. ad-hoc sítích, ve kterých lze pro zabezpečení použít pouze WEP.

## WPA2

implementuje všechny povinné prvky IEEE 802.11i. Konkrétně přidává k TKIP a algoritmu Michael nový algoritmus CCMP (*Counter Mode with Cipher Block Chaining Message Authentication Code Protocol*) založený na AES, který je považován za zcela bezpečný. Od 13. března 2006 je certifikace WPA2 povinná pro všechna nová zařízení, jež chtějí být certifikována jako Wi-Fi.

Některá zařízení podporují tzv. *Mixed mode*, kdy WLAN Access Point podporuje WPA a současně i WPA2 na jediném WLAN rozhraní, v některých případech umožňuje vytvořit i více, typicky 2 SSID buď přímo pro klienty, v jiných případech např. pro WDS režim, kdy páteř je šifrována nezávisle na klientském signálu, i když obě SSID, páteřní i klientské, musí být shodné, tak každé může mít jinou metodu šifrování či klíče.

**Advanced Encryption Standard (AES)** je symetrická bloková šifra (pro šifrování i dešifrování využívá stejný klíč na data s pevně danou délkou bloku), která nahradila dříve užívanou šifru DES.

## SSID

(Service Set Identifier) je jedinečný identifikátor každé bezdrátové (WiFi) počítačové sítě. Přístupový bod (AP) vysílá pravidelně každých několik sekund svůj identifikátor v takzvaném majákovém rámci (beacon frame) a klienti si tak mohou snadno vybrat, ke které bezdrátové síti se připojí. Parametr SSID se skládá z řetězce ASCII znaků dlouhého maximálně 32 znaků. Tento parametr představuje klíč, kterým dochází ke spojení jednotlivých adaptérů v rámci bezdrátové sítě. Všechna bezdrátová zařízení pokoušející se o vzájemnou komuni-

kaci mezi sebou musí předávat ten samý SSID. Pokud klíč klientského adaptéru se neshoduje s klíčem přístupového bodu (AP), je mu odmítnut přístup, proto se musí nastavit klíč shodně na přístupovém bodu (AP) a na klientském adaptéru. Nastavením různých klíčů můžeme zajistit fungování několika bezdrátových sítí v jedné lokalitě a v rámci stejného frekvenčního rozsahu. **SSID na bezdrátových klientech může být nastaven buď manuálně, vstupem SSID do klientského síťového nastavení, nebo automaticky.** Název sítě je obvykle nastaven k pojmenování síťového operátora, jako je například jméno společnosti. Hodnotu parametru SSID v síti je třeba považovat za první úroveň zabezpečení. Ve své základní, tovární podobě nemusí SSID poskytovat žádnou ochranu proti neoprávněnému přístupu k síti, pokud změníme na obtížněji rozluštitelný text, nedostanou se sem nekompetentní osoby tak snadno. **Je-li SSID v bezdrátové síti roven některé z výchozích hodnot, uvedený v tabulce, okamžitě jej změňte.** Tabulka výchozích hodnot SSID v bezdrátových sítích

Výrobce	Výchozí hodnota SSID
3Com	101, comcomcom
Cisco	Tsunami, WaveLAN Network
Compaq	Compaq
Dlink	WLAN
Intel	101, 195, xlan , intel
Linksys	Linksys, wireless
NetGear	Wireless
SMC	WLAN, SMC
Symbol	101
Teletronics	any
Vigor	default
Zcomax	any
Zyxel	Wireless, ZyXel
Ostatní	Wireless
CC&C	MyWlan

### Typy vysílání SSID

**Standardní SSID** - Správce sítě často používá veřejný SSID, který je nastaven na přístupný bod a vysílá ke všem bezdrátovým zařízením ve svém rozsahu.

**Nevysílající SSID** (not broadcasting SSID) Většina přístupových bodů (AP) v základní konfiguraci posílá informaci o své skupině SSID. **Pro větší bezpečnost se vypíná vysílání SSID.** V závislosti na bezdrátovém softwaru se uživateli síť buď neukazuje, nebo je zobrazená jako nepojmenovaná síť.

V každém případě, někdo potřebuje manuálně vstoupit do správy SSID pro připojení k síti. Tato metoda není bezpečná, protože pokaždé, když se někdo připojí k síti, odesílá SSID otevřeným textem, i když síťové spojení je jinak zašifrované. Bezpečnostní experti považují **vypínání SSID (pokud je to jediné bezpečnostní opatření proti zásahu třetích osob) za bezpečnostní slabinu.** Měly by být používány i další druhy šifrování a identifikace – WEP a nebo WPA.

Současné novější bezdrátové přístupové body zablokují automatickou SSID vysílací vlastnost v pokusu zlepšit síťové zabezpečení. Pokročilé bezdrátové přístupové body (AP) podporují vysílání mnohanásobných SSIDů, poskytujících vytváření virtuálních přístupových bodů (AP) - oddělení jednoho fyzického přístupového bodu (AP) do několika logických přístupových bodů (AP), kdy každý z nich může mít různý soubor bezpečnosti a síťového nastavení.

# Směrovače (routery) v sítích WAN

Routery jsou specifickým typem počítačů, obsahují stejné základní komponenty jako standardní PC ale vykonávají specifickou funkci.

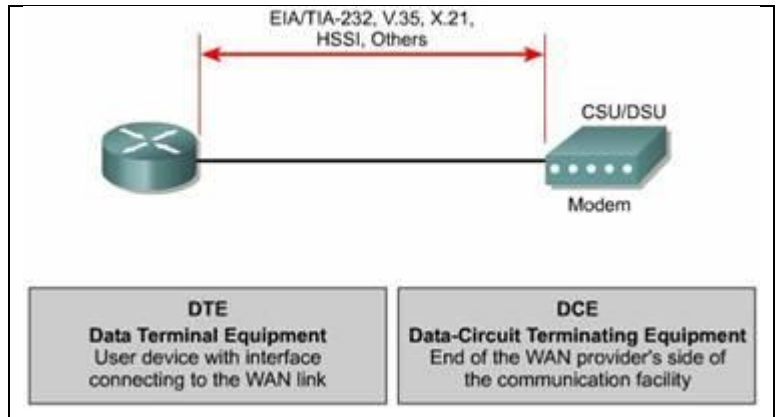
Routery obsahují: CPU, paměť, systémovou sběrnici a různá vstupně – výstupní rozhraní

Funkce: propojují sítě, určují nejlepší cestu dat

Routery mají WAN i LAN rozhraní. Pracují na 3. vrstvě OSI modelu, vytvářejí rozhodnutí na základě síťové (IP) adresy. Funkcí routeru je stavba směrovacích tabulek, výměna síťových informací s ostatními routery, určování nejlepší cesty při přepínání rámce.

## Funkce routeru ve WAN sítích.

Standardní protokoly používané ve WAN na 1. a 2. vrstvě jsou jiné než v LAN na stejné vrstvě. Fyzická vrstva popisuje propojení mezi data terminal equipment (DTE) a data circuit-terminating equipment (DCE). Zpravidla DCE je poskytovatel služby a DTE je zařízení, které se připojuje. DTE je schopné komunikovat jen s DCE zařízením, DCE určuje hodiny. Jednou z úloh routeru ve WAN je směrování paketů na 3. vrstvě, to je stejné jako v LAN.



Když router používá fyzickou a linkovou vrstvu s protokoly, které jsou spojené s WAN, pracuje jako WAN zařízení.

Primární úlohou routeru je poskytovat propojení na a mezi různými WAN fyzickými a linkovými standardy.

## Fyzická vrstva sítí WAN

### - WAN fyzické standardy:

- EIA/TIA-232
- EIA/TIA-449
- V.24
- V.35
- X.21
- G.703
- EIA-530
- ISDN
- T1, T3, E1, and E3
- xDSL
- SONET (OC-3, OC-12, OC-48, OC-192)

### - WAN linkové protokoly a standardy:

- High-level data link control (HDLC)
- Frame Relay
- Point-to-Point Protocol (PPP)
- Synchronous Data Link Control (SDLC)
- Serial Line Internet Protocol (SLIP)
- X.25
- ATM
- LAPB



- LAPD
- LAPF

## Vnitřní komponenty routeru

Komponenty routeru:

- CPU – The Central Processing Unit, vykonává instrukce v operačním systému, je to mikroprocesor, větší routery mohou mít více CPU
- RAM – Random-access memory, je používána pro směrovací tabulky, rychlou přepínací cache, směrovací konfigurace, vyrovnávací paměť pro IOS, při výpadku napájení se smaže, rozšiřuje se přes DIMMy.
- FLASH, pro uložení CISCO IOS (obrazu OS). Router normálně načítá IOS z FLASH paměti. Obraz OS může být obnoven nahráním novější verze. Obraz OS může být komprimovaný nebo nekomprimovaný. Přidáním nebo nahrazením flash paměti SIMMy nebo PCMCIA kartou, se dá rozšířit velikost flash paměti.
- NVRAM – Nonvolatile random-access memory, používaná na uložení spouštěcí konfigurace, vypnutím sa nemaže.
- Zbernice, delíme na systémovú a CPU zbernici. Systémová slúži na komunikáciu medzi CPU a rozhraniami, rozširujúcimi slotmi. CPU zbernica komunikuje z pamäťami.
- ROM – Read-only memory, je použitá pre trvalé uloženie spúšťacieho diagnostického kódu. Hlavná úloha je hardverová diagnostika počas spustenia routera a nahranie IOS z flešky do RAM. Niektoré routery majú obmedzenú verziu IOS, ktorá môže byť použitá ako alternatívny bootovací zdroj. ROM sa nedá zmazať. Upgrade sa robí výmenou čipu.

- rozhraní,

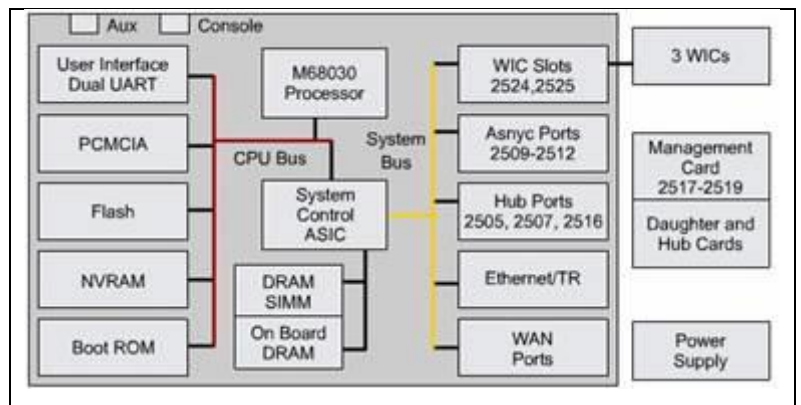
**LAN, Ethernet alebo Token-Ring**

**WAN, sériové, ISDN, integrované CSU**

(LAN i WAN rozhraní buď pevná konfigurace nebo modulární)

**Console/AUX, sériový, na inicializaci routeru, (AUX pro připojení modemu)**

- napájecí zdroj, napája všetky komponenty routera



## Console port a konektorové piny

Použijeme RJ-45-na-RJ-45 rollover kabel a RJ-45-na-DB-9 female DTE adaptér (označený TERMINAL) k připojení console portu do PC, s běžícím emulačním terminálovým softwarem (Hyperterminal). Tabulka zobrazuje signály a piny asynchronního sériového console portu, RJ-45-na-RJ-45 rollover kabelu, a RJ-45-na-DB-9 female DTE adaptéru (označeného TERMINAL).

Tabulka signálů Console portu, kabelů a DB-9 adaptéru				
Console port (DTE)	RJ-45-na-RJ-45 Rollover kabel		RJ-45-na-DB-9 Terminal adaptér	PC konzole
Signál	RJ-45 Pin	RJ-45 Pin	DB-9 Pin	Signál
RTS	1 <sup>1)</sup>	8	8	CTS
DTR	2	7	6	DSR
TxD	3	6	2	RxD
GND	4	5	5	GND

GND	5	4	5	GND
RxD	6	3	3	TxD
DSR	7	2	4	DTR
CTS	8 <sup>1)</sup>	1	7	RTS

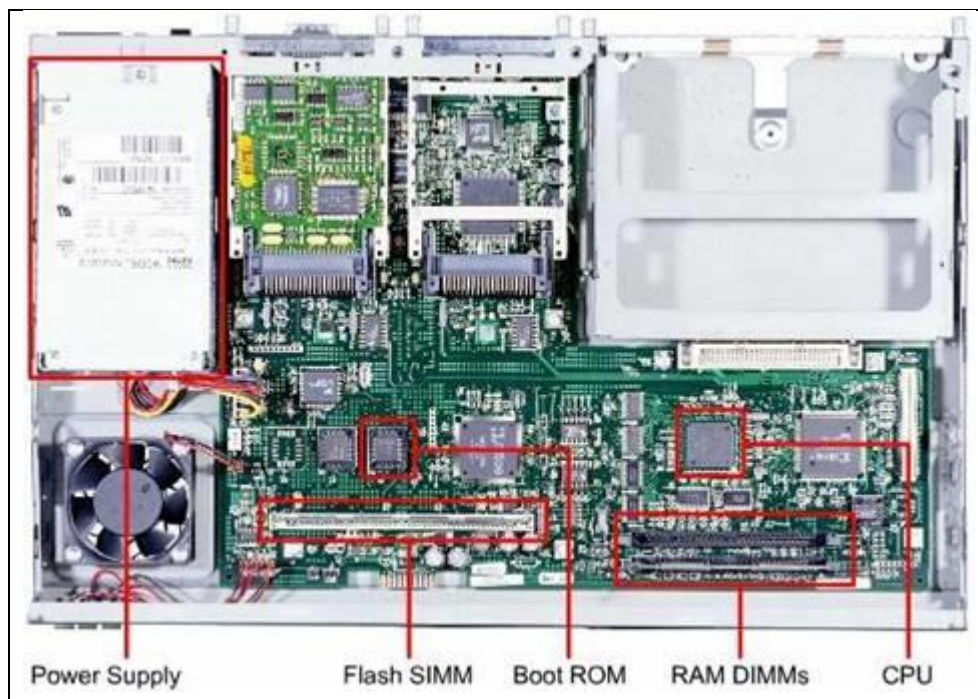
<sup>1)</sup> Pin 1 je připojen uvnitř na pin 8.

### Auxiliary port a konektorové piny

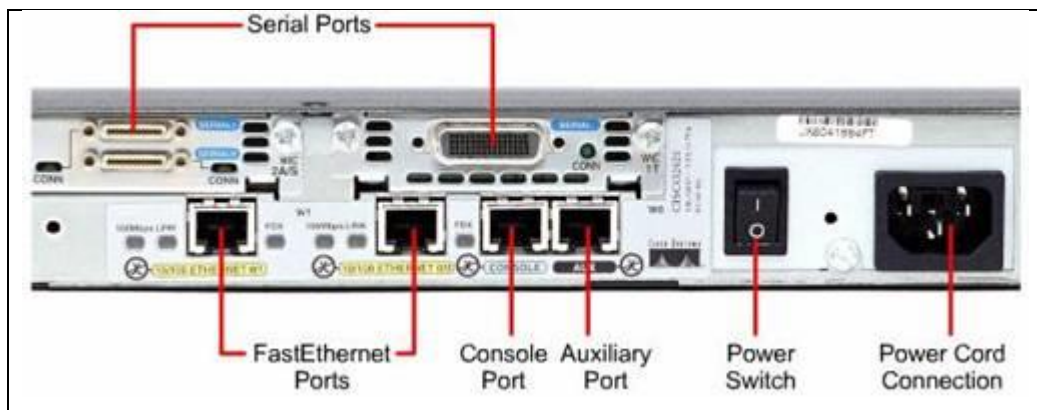
Použijeme **RJ-45-na-RJ-45 rollover kabel** a **RJ-45-na-DB-25 male DCE adaptér** (označený MODEM) k připojení auxiliary portu na modem. Tabulka zobrazuje signály a piny asynchronního sériového auxiliary portu, RJ-45-RJ-45 rollover kabelu, a RJ-45-DB-25 male DCE adaptéru (označeného MODEM).

Tabulka signálů Auxiliary portu, kabelů a DB-25 adaptéru				
AUX Port (DTE)	RJ-45-na-RJ-45 Rollover Cable		RJ-45-na-DB-25 Modem Adapter	Modem (DCE)
Signál	RJ-45 Pin	RJ-45 Pin	DB-25 Pin	Signál
RTS	1	8	4	RTS
DTR	2	7	20	DTR
TxD	3	6	3	TxD
GND	4	5	7	GND
GND	5	4	7	GND
RxD	6	3	2	RxD
DSR	7	2	8	DCD
CTS	8	1	5	CTS

Vnitřní uspořádání routeru:



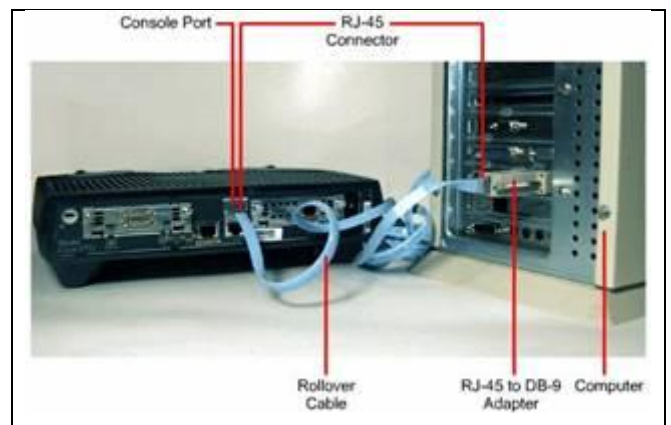
Zadní strana routeru – uspořádání konektorů:



## Připojení směrovače ke konzole

Řídící porty:

- **konzolový port**, doporučený pro inicializaci, připojuje se na sériový port PC
  - **AUX**, pro dálkovou správu přes modem
- Nejsou navrženy jako síťové porty, jeden z nich je potřebný pro inicializační konfiguraci routeru. **Po prvním zapnutí není router nakonfigurovaný**, proto se připojuje přes RS232 k ASCII terminálu nebo počítačovému terminálu. Konzolový port může být též použitý při řešení problémů, router vypisuje všechny úkony při spouštění, případně při změně hesla ke správě.



## Připojení konzolového rozhraní

Připojení je realizované přes konzolový port, rollover kábel, obsahující konektor RJ45 a DB9.

Terminál musí podporovat VT100 emulaci.

Připojení na router:

### 1. konfigurace terminálu:

- přidělit com port (COM1,...)
- nastavit rychlost přenosu 9600 baud
- nastavit 8 datových bitů
- nastavit žádná parita
- nastavit 1 stop bit
- nastavit žádné řízení toku

2. připojit konektor RJ45 rollover kabelu do routeru na konzolový port
3. (připojit konektor RJ45 adaptéru na rollover kabel)
4. zapojit DB9 konektor (adaptéru) do PC

## 1.2.6. Připojenia LAN rozhrania

Router komunikuje z LAN cez HUB alebo SWITCH. Na pripojenie sa používa priami kábel. V niektorých typoch Ethernetového prepojenie je router pripojený priamo na PC alebo na iný router, vtedy

sa používá překřížený kabel.

Pro lepší rozlišení portů v CISCO zavedli barevné rozlišení kabelů. (viz tabulka)

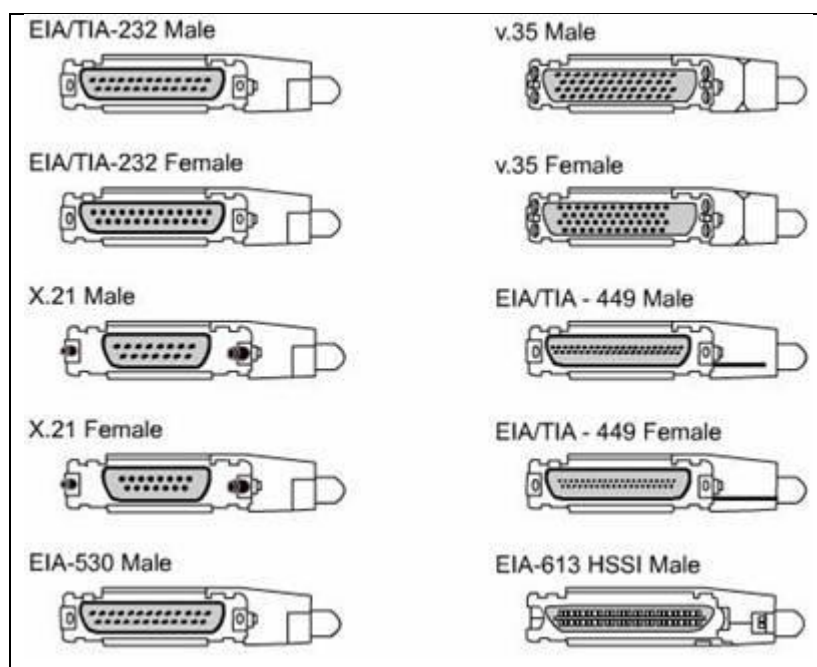
## Připojení WAN rozhraní

WAN připojení je realizované na velké vzdálenosti různými typy technologií. Tyto WAN služby jsou vždy pronajaté od servisního poskytovatele.

Pro každý typ WAN služby, customer premises equipment (CPE), často router, je data terminal equipment (DTE). Na poskytovatele služeb se připájame cez DTE, často je top mode.

Port or Connection	Port Type	Color	Connected To	Cable
Ethernet	RJ-45	yellow	Ethernet hub or Ethernet switch	Straight-through
T1/E1 WAN	RJ-48C/CA81A	light green	T1 or E1 network	RJ-48 T1
Console	8 pin	light blue	Computer com port	Rollover
AUX	8 pin	black	Modem	Rollover
BRI S/T	RJ-48C/CA81A	orange	NT1 device or private integrated network exchange (PINX)	RJ-48
BRI U WAN	RJ-49C/CA11A	orange	ISDN network	RJ-49
Token	UTP, STP	purple	Token Ring device	RJ-45 Token Ring cable

DCE sériová rozhraní:



## Operační systém IOS

Internetwork Operating System software (**IOS**) – operační systém routeru, slouží na spuštění konfiguračních souborů.

Konfigurační soubory: obsahují veškeré informace pro nastavení a použití, povolení směrování a směrovací protokoly v routeru.

### Hlavní vnitřní komponenty směrovače:

- random access memory (RAM)
- nonvolatile random-access memory (NVRAM)
- flash memory
- read-only memory (ROM)
- rozhraní

Funkce **RAM**: pracovní prostor pro OS, směrovací tabulky, běžící konfigurace, vyrovnávací paměti

Funkce **NVRAM**: startovní konfigurace (vypnutím se neztratí)

Funkce **FLASH**: kompletní IOS, aktualizovatelný, též více verzí současně

Funkce **ROM**: diagnostické a zaváděcí (bootstrap) programy (ROM monitor), obvykle obsahuje také omezenou verzi IOSu, nedá se aktualizovat

Sběrnice: system bus a CPU bus, vnější rozhraní

## Účel Cisco OS IOS

Bez operačního systému nemá router žádné schopnosti.

Služby Cisco IOSu:

1. základné smerovacie a prepínacie funkcie
2. spoľahlivý a bezpečný prístup na sieťové zdroje
3. **síťová škálovateľnosť** - (**Škálovateľnosť** neboli rozšiřitelnost v telekomunikáciách a softwarovém inženýrství je žádoucí vlastnost systému, síť nebo procesu, která má schopnost pracovat s náhlými změnami potřeby obsluhy čili zvyšovat sledované parametry v případě, že nastane taková potřeba.)

## Uživatelské rozhraní routeru

Cisco IOS používá **command-line interface (CLI)** jako tradiční konzolové prostředí, které je dostupné několika způsoby:

- **webové rozhraní**
- **console** případně AUX pomocí modemu. Používá se nízkorychlostní sériové připojení přímo z PC na router.
- **Telnet** jedno z rozhraní musí být nakonfigurované s IP adresou, a virtuální připojení musí mít nakonfigurované jméno a heslo

## Módy uživatelského rozhraní routeru

OS IOS pracuje ve dvou módech. Každý mód je indikován jiným promptem a umožňuje jen příkazy přidělené tomuto módu. Cisco pracuje na vývoji různých typů IOSu, každý je určený jen pro dané zařízení, avšak zachovává základní konfigurační strukturu.

Command executive (EXEC) – IOS poskytuje command interpreter service, ten schvaluje a vykonává příkazy

Rozdělení EXEC podle bezpečnosti:

- **uživatelský EXEC mód**

umožňuje jen základní počet monitorovacích příkazů, jen prehliadanie, identifikácia: ">"

- **privilegovaný EXEC mód**, tiež známi ako povolený mód

umožňuje vykonávať všetky príkazy routera, je možné nastaviť heslo pre vstup do tohto módu, používa sa pre konfigurácia, identifikácia: "#"

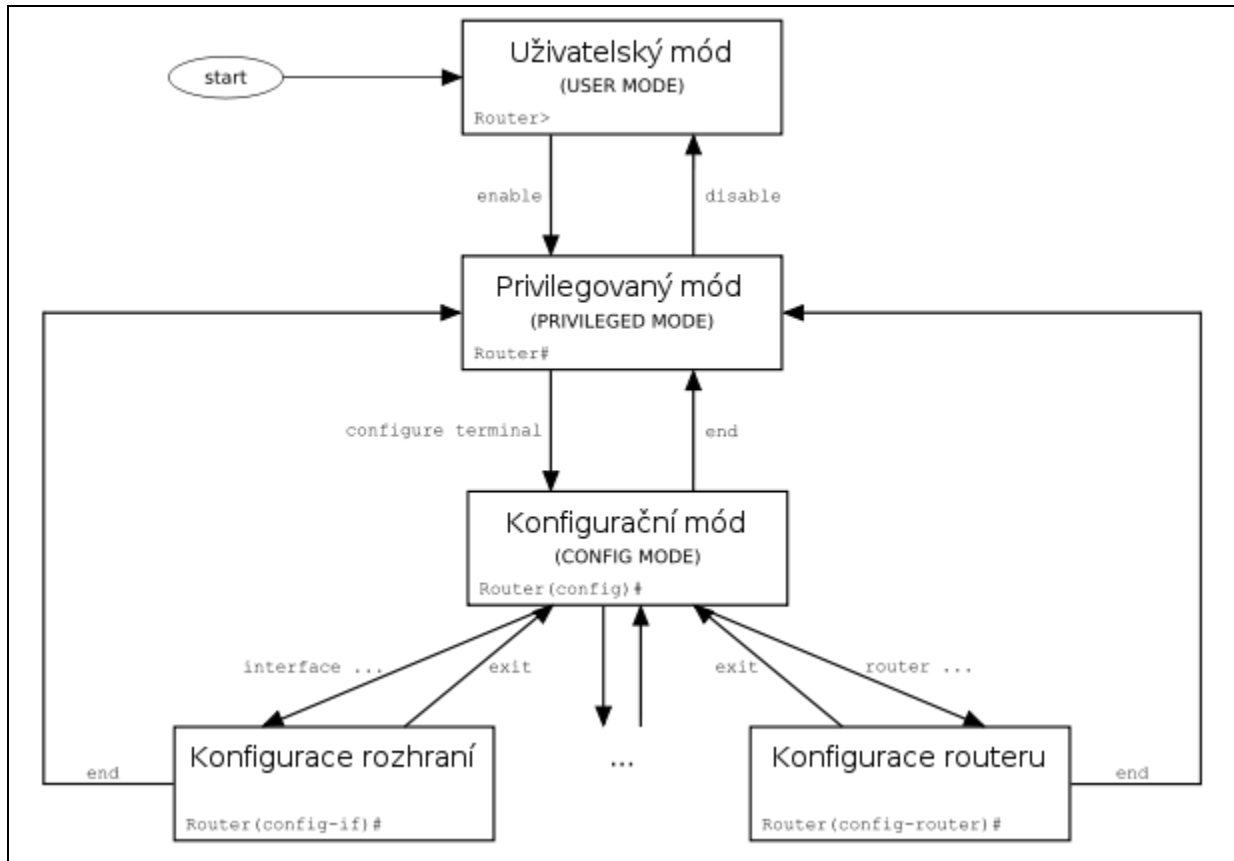
**Enable** – príkaz na prechod z užívateľského módu do privilegovaného, prejaví sa zmenou promptu

**Exit** – ukončenie privilegovaného módu

**?** – provede výpis príkazů pro jednotlivé módy, nebo help k příkazu



Základní módy a příkazy, kterými lze mezi těmito módy přecházet:



### Klávesové zkratky

- Šipka nahoru (↑) nebo **Ctrl+P** - zobrazí předchozí příkaz
- Šipka dolů (↓) nebo **Ctrl+N** - zobrazí následující příkaz
- Šipka doprava (→) nebo **Ctrl+F** - posun o jeden znak vpřed
- Šipka doleva (←) nebo **Ctrl+B** - posun o jeden znak zpět
- **Tab** - automatické doplnění příkazu

### Boot (zavedení) Cisco IOS OS

Stavy po zavedení Cisco IOS OS:

1. **režim ROM monitor** – zobrazí se výpis spouštěcího procesu, základní funkce a diagnostika, **identifikace promptem ">"**. Používá se na obnovu OS při systémovém selhání a při obnově hesla, **do tohoto módu je možné vstoupit jen přes konzolový port**
2. **omezená verze Cisco IOS OS** – zavedení proběhne z ROM, je určená na výměnu obrazu Cisco IOSu uloženého ve FLASH paměti, **identifikace promptem "(boot)>"**. Příkaz: **copy tftp flash** – příkaz na obnovu IOSu, uloženého na TFTP serveru, do FLASH paměti v routeru. Při výběru nového IOS obrazu je důležitá kompatibilita s FLASH a RAM pamětí routeru.
3. **IOS je zaváděn z FLASH** - normální provoz routeru, **identifikace promptem "Router>"**

Bootstrap proces normálně nahrává do RAM jedno z těchto prostředí na základě nastavení konfiguračního registru.

### Inicializačné spustenie CISCO routra

Po zapnutí routra sa spustí nahrávanie bootstrap, nasleduje operačný systém a konfiguračný súbor. Ak nevie nájsť konfiguračný súbor, zavedie sa setup mód. Primárne cieľom je spustiť smerovacie operácie. Na zavedenie tohto cieľu musí:

1. **uistiť sa o funkčnosti hardwaru a otestovať ho**
2. **nájsť a spustiť Cisco IOS software**

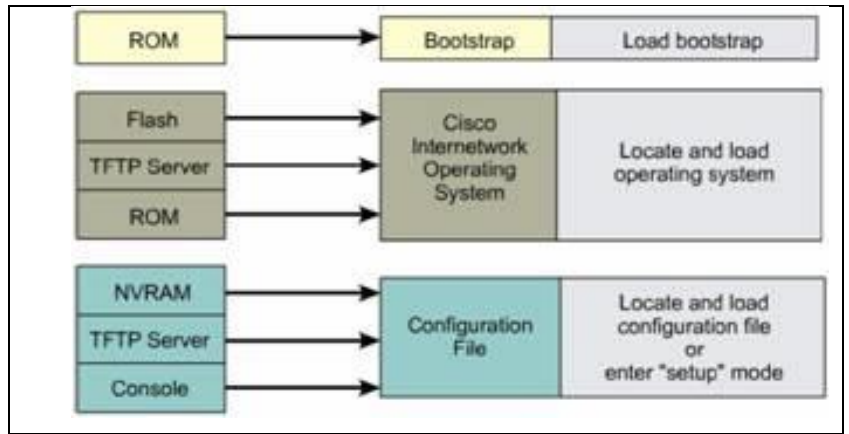


### 3. nájsť a aplikovať konfiguračný súbor alebo sprístupniť setup mode

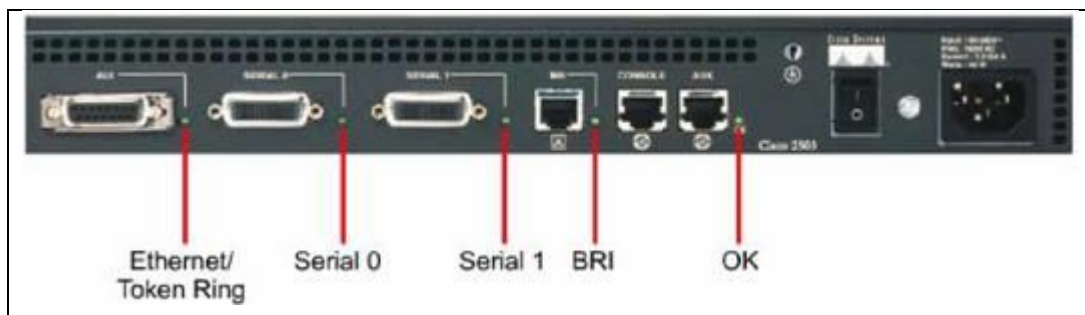
Po zapnutí routra sa vykonáva power-on self test (POST), diagnostika základných funkcií CPU, pamäte a sieťových portov. Po tomto teste sa prevádza software inicializácie.

Po POST probíhá:

1. spustí sa bootstrap, súbor inštrukcií testujúcich hardware a inicializácia IOS
2. IOS môže byť na viacerých miestach, rozhoduje sa na základe konfiguračného registru, môže byť vo FLASH alebo nahraný zo siete. V konfiguračnom súbore je udaná cesta a meno image.
3. nahranie image IOS, po funkčnom zavedení sa vypisujú pripravený hardware a software na konzole terminálu
4. konfiguračný súbor je nahraný z NVRAM do RAM, spúšťa sa smerovací proces
5. ak nebola nájdená konfigurácia v NVRAM IOS hľadá konfiguráciu na TFTP serveri. Ak nie je nájdená konfiguračný súbor, spustí sa setup dialóg.



V setup móde sa nás pýta, či nahráť default konfiguráciu. Ukončenie sprievodcu je možné cez **Ctrl+C** LED indikujú aktivitu na príslušnom rozhraní. Ak zelená OK LED svieti, systém je zavedený správne



### Skúmanie inicializačného bootup routra

Po 1. spustení nie je v NVRAM žiadny konfiguračný súbor, preto je zobrazené hlásenie "NVRAM invalid, possibly due to write erase", router musí byť skonfigurovaný a konfigurácia uložená do NVRAM konfiguračného súboru.

Konfiguračný register je od výroby nastavený na 0x2102, konfiguračný súbor sa nahráva z FLASH pamäte.

Z výpisu na konzole je možné určiť:

1. bootstrap verziu a IOS verziu
2. model routra
3. procesor
4. hodnota celkovej pamäte
5. počet a typ rozhraní
6. veľkosť NVRAM a FLASH pamäte

**Show version** – kontrola aktuálneho image a dostupnej pamäte RAM, zobrazí verziu, to čo pri štarte, či bútuje z ROM alebo FLASH

príklad: ... <output omitted>... cisco 1721 (68380) processor (revision C) with 3584K/512K bytes of memory.

c4500-i-mz\_112-16.bin

1. platforma c4500

2. internetový i
3. komprimovaný mz
4. verzia 11.2(16)

**Show flash** – zistenie kapacity flash pamäte, verifikácia dostatku miesta pre nový IOS

príklad: ...15998976 bytes total (10889728 bytes free)

#### 2.2.4. Zavedenie HyperTerminal spojenia

Všetky Cisco routre obsahujú asynchrónny TIA/EIA-232 sériový konzolový port (RJ-45). Na vytvorenie spojenia sú potrebné káble a adaptér. Konzolový terminál je ASCII terminál alebo PC, na ktorom je spustený emulátor terminálu napr. HyperTerminal. Postup pripojenia:

1. pripojíme jeden koniec rollover kábla k routru, druhý pripojíme cez redukciu BD-9 alebo 25 k COM1 / 2.
2. skonfigurujeme terminál 9600baud, 8 data bits, bez parity, 1 stop bit, a bez kontroly toku

#### 2.2.5. Prihlásenie do routra

Na konfiguráciu je potrebné mať prístup cez terminál alebo cez diaľkový prístup, pri vstupe sa užívateľ musí prihlásiť.

Úrovne bezpečnosti:

- Užívateľský EXEC mód, nie je dovolená konfigurácia routra, len prezeranie, príkazy sú podmnožinou privilegovaného módu
- Privilegovaný EXEC mód, umožňuje zmeny s konfiguráciou

Na sprístupnenie privilegovaného módu sa zadáva príkaz **Enable**, môže byť vyžadované heslo. Zmena promptu z > na #

**Nastavení hesla pro přístup.**

**enable password** slabo **enable secret**: nastavenie hesla do privilegovaného módu

**disable** alebo **exit** alebo **Ctrl+Z** návrat do užívateľského módu

#### 2.2.6. Kláves help v routri CLI

**?** – výpis príkazov podľa módu,

pri zobrazení "--More--" sú príkazy ešte aj na ďalšej obrazovke, **mezerníkem prejdeme o obrazovku ďalej, o riadok ďalej cez Enter.**

Na prístup do privilegovaného módu je stačí zadat **ena**, což je zkratka **enable**.

Príklad využití helpu na nastavení hodin:

1. predpokladáme, že nepoznáme správny príkaz, zadáme **?**, medzi príkazmi je **clock**
2. potrebujete zistiť syntax, zadáme **clock ?**
3. clock set 15:38:50 12 jan 2004 a potvrdíme Enter Šípkami hore dole sa vypisuje história, je nastavený na 20, dá sa meniť

**Názvová konvencia súborov s Cisco IOS:**

**c2600 - js - 1 . 122-12.bin**

**122-12** - typ verzie(12.212)

**1** - súborový formát (nekomprimovaný)

**js** - (podnikový s rozšírenou kapacitou)

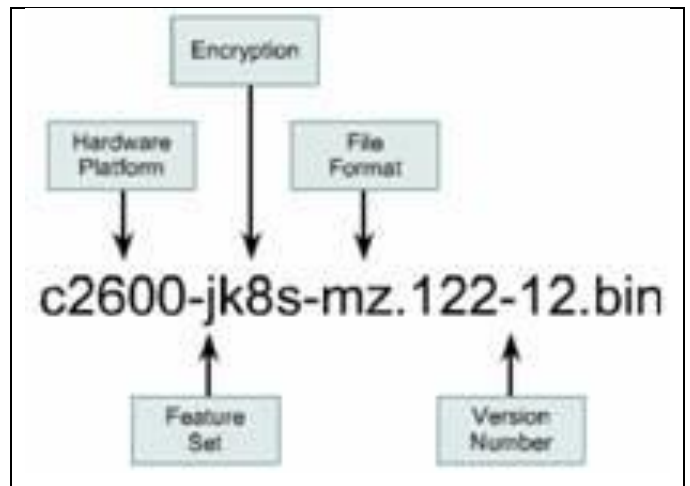
**c2600** - hardwarová platforma (2600)

- na odlíšenie rôznych verzií

### Příkaz show version

Zobrazí informácie o Cisco IOS software verzii bežiac na routri. Obsahuje nasledujúce:

1. IOS verzia a popisné informácie
2. Bootstrap ROM verzia
3. Boot ROM verzia
4. Doba, ako dlho router beží
5. posledná metóda reštartu
6. image systémového súboru a jeho umiestnenie
7. platformu routra
8. nastavenie konfiguračného registra



## Konfigurování routeru

### 3.1.1. CLI príkazové módy

Všetky konfiguračné zmeny Cisco routra sa robia v globálnom konfiguračnom móde. Sprístupnenie GKM: cez skratku **conf t** alebo

**Router#configure terminal**

**Router(config)#** globálny konfiguračný mód GKM obsahuje ešte ďalšie módy:

- |                      |                                   |
|----------------------|-----------------------------------|
| 1. interface mode    | <b>Router(config-if)#</b>         |
| 2. line mode         | <b>Router(config-line)#</b>       |
| 3. router mode       | <b>Router(config-router)#</b>     |
| 4. subinterface mode | <b>Router(config-subif)#</b>      |
| 5. controller mode   | <b>Router(config-controller)#</b> |

jsou ještě další

Všetky zmeny, ktoré sa vykonávajú v jednotlivých módoch, sú aplikované len na rozhranie alebo proces, ktorý danému módu zodpovedá.

**exit** – návrat ze specifického konfiguračného módu do globálneho konfiguračného módu

**Ctrl+Z** – ukončenie konfiguračného módu a návrat do privilegovaného módu

### Konfigurování jména routeru

Změna jména routeru:

**Router(config)#**

**hostname Tokyo**

**Tokyo(config)#**

### Příkazy show:

**show interfaces** – zobrazí všetky štatistiky pre všetky rozhrania routra, ak nás zaujíma špecifické rozhranie, stačí zadať: **Router#show interfaces serial 0/1**

**show controllers serial** – zobrazí špecifické informácie o rozhraní hardwaru

**show clock** – zobrazí čas nastavený v routri

**show hosts** – zobrazí kešovaný list hostiteľského mena a adresy

**show users** – zobrazí všetkých pripojených užívateľov k routru

**show history** – zobrazí históriu príkazov

**show flash** – zobrazí informáciu o flash pamäti a ktoré IOS súbory sú v nej uložené

**show version** – zobrazí informáciu o routri a IOS, ktorý beží v RAM

**show ARP** – zobrazí ARP tabuľku routra

**show protocol** – zobrazí globálne a rozhranové informácie týkajúce sa všetkých konfigurovaných L3 protokolov

**show startup-configuration** – zobrazí uloženú konfiguráciu v NVRAM

**show running-configuration** – zobrazí konfiguráciu aktuálne bežiacu v RAM

### Konfigurovanie sériového rozhrania

Môže byť konfigurované cez konzolu alebo cez virtual terminal line.

1. vstup do globálneho konfiguračného módu
2. vstup do módu rozhrania
3. uviesť adresu pre rozhranie a masku
4. nastaviť clock rate ak ide o DCE
5. zapnúť rozhranie

Každé pripojené sériové rozhranie musí mať IP adresu a masku siete.

Príklad:

```
Router(config)#interface serial 0/0
```

```
Router(config-if)#ip address <ip address> <netmask>
```

(Sériové rozhranie vyžaduje hodinový signál na kontrolu časovania komunikácie. DCE zariadenie treba nastaviť na nastavenie poskytovania hodinového signálu.)

```
Router(config-if)#clock rate – spúšťa časovač a definuje rýchlosť: 1200, 2400, 9600, 19200, 38400, 56000, 64000, 72000, 125000, 148000, 500000, 800000, 1000000, 1300000, 2000000, alebo 4000000.
```

```
Router(config-if)#no shutdown – zapína rozhranie
```

**shutdown** – vypína rozhranie

Príklad využitia príkazov pri rýchlosti 56000:

```
Router(config)#interface serial0/0
```

```
Router(config-if)#ip address 192.168.2.1 255.255.255.0
```

```
Router(config-if)#clock rate 56000
```

```
Router(config-if)#no shutdown
```

### Přidání, přesun a změny konfigurace

Ak konfigurácia vyžaduje konfiguráciu, treba prejsť do potrebného módu, napríklad ak potrebujem zapnúť nejaké rozhranie, spustím globálny konfiguračný mód, vstúpim do módu daného rozhrania a spustím ho cez príkaz **no shutdown**.

**show running-config** – overenie aktuálnej konfigurácie, je čítaná z RAM

Postup, ak zmeny nerobia požadované:

1. Zrušenie/zmena konfiguračného príkazu v príslušnom konfiguračnom móde!!!–no príkaz(no ip address, no shutdown)
2. obnovenie konfigurácie z NVRAM
3. skopírovať konfiguráciu cez TFTP

**erase startup-config** – odstráni spúšťaciu konfiguráciu z NVRAM, po reštarte prejsť do setup módu

**copy running-config startup-config** – uloží bežiacu konfiguráciu do NVRAM

### Konfigurovanie Ethernetového rozhrania

Je možné konfigurovať cez konzolu alebo cez virtual terminal line. Každé Ethernetové rozhranie musí mať IP a masku. Postup konfigurácie:

1. spustiť globálny konfiguračný mód
2. spustiť konfiguračný mód rozhrania
3. špecifikovať IP a masku
4. povoliť rozhranie Rozhranie je prednastavené ako vypnuté, spustíme ho cez **no shutdown**, vypneme cez **shutdown**.

#### 3.2.6. Preklad hostiteľských mien

V tomto procese sa asociujú symbolické mená z IP adresou.

```
Router(config)#ip host Auckland 172.16.32.1
Router(config)#ip host Beirut 192.168.53.1
Router(config)#ip host Capetown 192.168.89.1
Router(config)#ip host Denver 10.202.8.1
```

Pri smerovačoch je vhodné uviesť IP adresy všetkých rozhraní.

Host table – list hostiteľských mien a ich asociovaných IP adries, routre si ju udržiavajú v keši, čím sa zvyšuje rýchlosť prevodu mien na adresy. Príklad konfigurácie:

#### 3.2.7. Konfigurovanie hostiteľskej tabuľky

Postup:

1. vstup do GKM v routri
2. zadám príkaz **ip host** meno routra a všetky IP adresy asociované z rozhraním routra
3. pokračovať v zadávaní, až sú všetky routre na sieti vložené
4. uložiť konfiguráciu do NVRAM V príkazoch telnet/ping/trace je možné použiť symbolické meno namiesto IP adresy

### Mapování okolních zařízení

#### Úvod do CDP

**Cisco Discovery Protocol** (CDP) – pracuje na 2 vrstve, prepája nižšie vrstvy protokolu s vyššími. Používa sa na získanie informácií o susedných zariadeniach bez ohľadu na konfigurovaný sieťový protokol, je možné získať typ pripojeného zariadenia, rozhranie, ktorým je pripojené a model zariadenia. Pracuje na všetkých zariadeniach CISCO a len s Cisco zariadeniami. Pri spustení zariadenia sa automaticky spúšťa aj CDP, každé zariadenie konfigurované pre CDP vysiela periodicky správu, obsahuje dobu platnosti.

#### Informácia získaná z CDP

Primárna funkcia CDP je získavanie informácií o všetkých priamo prepojených susedných CISCO zariadeniach.

**show cdp neighbors** : zobrazí všetkých priamo pripojených susedov Pomocou tohto príkazu je možné zistiť:

1. Device ID (meno vzdialeného zariadenia)
2. Local Interface (cez ktoré naše lokálne rozhranie je dostupné)
3. Holdtime(kedy vyprší platnosť informácie–implicitne sú CDP pakety vysiellané každých 60 s. a platnosť informácie je 180 s.)
4. Capability(smerovač, prepínač, ...)
5. Platform(konkrétna platforma zariadenia)
6. Port ID (port na vzdialenom zariadení ,ktorý nám vyslal informáciu)
7. VTP Management Domain Name (CDPv2only)
8. Native VLAN (CDPv2only)

## 9. Full/Half-Duplex (CDPv2only)

### 4.1.3. Realizovanie, monitorovanie a údržba CDP

**cdp run** – globálne povolenie CDP na routri, býva defaultne povolené, doporučuje sa však zakázať kvôli bezpečnosti, GKM

**cdp enable** – povolenie CDP na určitom rozhraní, interface KM

**clear cdp counters** – vynuluje CDP počítadlá, užívateľský mód

**show cdp** – zobrazí interval medzi prenosom Z CDP zariadení, počet sekúnd CDP správy platnej na danom porte a verziu správy, privilegovaný mód

**show cdp entry „meno zariadenia“** - zobrazí informáciu o špecifickom susedovi, zobrazenie môže byť limitované protokolom alebo verziou správy, rozlišujú sa malé a veľké písmená v názve, ak je \*, zobrazia sa všetky zariadenia, privilegovaný mód

**show cdp interface „typ a číslo“** – zobrazí informáciu o rozhraní, na ktorom je CDP povolené, privilegovaný M

**show cdp neighbors** - zoznam susedov zistený cez CDP, privilegovaný mód

**show cdp traffic** - overenie, či som skutočne čosi poslal, užívateľský mód

### 4.1.4. Vytváranie sieťovej mapy okolia

Hoci sú CDP rámce malé, poskytujú detailné informácie o pripojených CISCO susedných zariadeniach. Pomocou týchto informácií môžeme vytvoriť mapu pripojených zariadení. Cez telnet je možné prihlásenie na tieto zariadenia a cez príkaz **show cdp neighbors** zobrazíť susedné zariadenia.

### 4.1.5. Zakázanie CDP

**no CDP run** – v GKM, zakáže CDP pre všetky individuálne rozhrania

Vo verzii 10.3 a vyššie je CDP povolené, je možné posilať a prijímať CDP informácie.

**no CDP enable** command – na zakázanie CDP na určitom rozhraní, ak CDP bolo povolené, interface KM

### 4.1.6. Odstraňovanie porúch CDP

**clear cdp table** – zmaže informácie o susedoch z CDP tabuľky

**clear cdp counters** – vynulovanie počítadiel o prijatý / vyslaných CDP paketoch

**show cdp traffic** – zobrazí CDP počítadlá, obsahujú počet prijatých a odoslaných CDP paketov a kontrolný súčet

**show debugging** – zobrazí informáciu o tipoch ladenia, ktoré sú povolené

**debug cdp adjacency** – CDP susedné informácie

**debug cdp events** – CDP udalosti

**debug cdp ip** – CDP IP informácie

**debug cdp packets** – informácie súvisiace s paketom

**cdp timer** – špecifikuje, ako často Cisco IOS posiela CDP aktualizácie

**cdp holdtime** – určuje dobu platnosti poslaného paketu

**show cdp** – zobrazí globálne CDP informácie obsahujúce časovač a dobu platnosti

## Smerovanie a smerovacie protokoly

### 6.1.1. Úvod do smerovania

Smerovanie je proces, ktorý router používa na presun paketov do cieľovej siete, vytvára rozhodnutie na základe cieľovej IP adresy, z paketu určí jeho ďalší smer, ktorý ho dovedie do cieľa. Postupy učenia routra na určenie správneho smerovania:

1. dynamické smerovanie, smerovacie informácie sa učí od iných routrov, pri veľkých sieťach
2. statické smerovanie, sieťový administrátor konfiguruje ručne informácie o smerovaní, malé siete

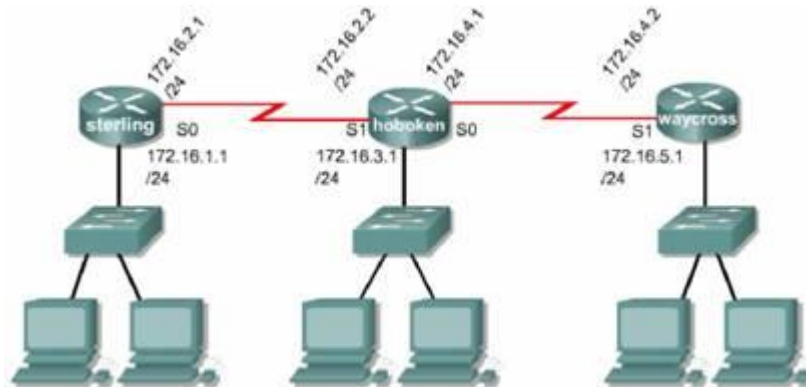


### 6.1.2. Statické smerovanie

#### Smerovacie operácie:

```
Hoboken(config)#ip route 172.16.1.0 255.255.255.0 s0
command destination net subnet mask outgoing
interface
```

1. sieťový administrátori konfigurujú routre
2. router inštaluje smer smerovania v smerovacej tabuľke
3. pakety sú smerované použitím statických pravidiel **ip route** – v GKM, definuje statické smerovanie routra



Administrátor Hoboken potrebuje prístup k sítim 172.16.1.0/24 a k 172.16.5.0/24. Sítě, ktoré jsou priamo pripojené **netreba** definovať. Môže to zrealizovať dvoma spôsobmi:

```
Hoboken(config)#ip route 172.16.1.0 255.255.255.0 172.16.2.1
```

```
Hoboken(config)#ip route 172.16.5.0 255.255.255.0 172.16.5.1 alebo
```

```
Hoboken(config)#ip route 172.16.1.0 255.255.255.0 s1
```

```
Hoboken(config)#ip route 172.16.5.0 255.255.255.0 s0
```

K tomuto príkazu je možné pridať ďalší parameter, administratívna vzdialenosť v rozsahu od 0-255, udáva mieru spoľahlivosti, čím menšie číslo, tým je spoľahlivosť vyššia. Default je 1. Vďaka tejto možnosti môžu byť v tabuľke dva rovnaké údaje, ale AV je iná, router dá prednosť záznamu s nižšou AV.

#### Konfigurácia prednastaveného smerovania

**Default route** je použitá na smerovanie paketov, ktorý sa nezhoduje ani z jedným smerom v smerovacej tabuľke. Umožňuje udržiavať menšie smerovacie tabuľky (internet), lebo nie všetky smery v sieti musia byť v nich explicitne uvedené

**ip route 0.0.0.0 0.0.0.0 nasledujúci smerovač a alebo rozhranie k nemu** : v GKM, smerovanie všetkých paketov, ktoré nie sú v smerovacej tabuľke cez špecifikované rozhranie alebo IP adresu nasledujúceho smerovača.

Postup konfigurácie:

1. spustiť GKM
2. zadať príkaz **ip route 0.0.0.0 0.0.0.0**, kde 0.0.0.0 0.0.0.0 sú cieľové IP a maska siete, cieľová brána je meno rozhrania alebo IP adresa nasledujúceho routra
3. odchod z GKM
4. uloženie aktuálnej konfigurácie: copy running-config startup-config

### 6.1.5. Overovanie konfigurácie statického smerovania

Po zadaní smerovanie treba overiť, že smerovanie je v smerovacej tabuľke a preveriť, či pracuje podľa očakávaní.

**show running-config:** overenie, že statické smerovanie bolo zadané správne **show ip route:** príkaz na overenie, že statické smerovanie je v smerovacej tabuľke Postup overovania:

1. show **running-config**, zadám v privilegovanom móde, vypíše aktívnu konfiguráciu
2. overím, že statické smerovanie bolo správne vložené, pri chybnom, v GKM odstránim chybný riadok a vložím správny, ku koncu sa zobrazí riadok: ip route IP maska rozhranie
3. zadám príkaz: **show ip route**
4. overím, že smerovanie, ktoré bolo vložené, je v smerovacej tabuľke

### Uživatelský mód (User Mode)

Mód, ve kterém se nacházíte po navázání relace s routerem/switchem. Neumožňuje měnit konfiguraci.

enable	Vstup do privilegovaného módu
ping [...]	Příkaz PING zjišťuje dostupnost cíle pomocí protokolu ICMP.
traceroute [...]	Trasování cesty k cíli pomocí protokolu ICMP.
exit	Příkaz ukončí relaci.
logout	Příkaz ukončí relaci.
show	Příkaz zobrazí nejružnější informace.
show flash	Příkaz zobrazí informace o vnitřní paměti FLASH.
show ip interface [...]	Příkaz zobrazí informace o fyzických rozhraních (např. Serial1/0), na kterých je nakonfigurován IP.
show ip route	Příkaz zobrazí routovací tabulku.
show version	Příkaz zobrazí informace o hardwaru, IOS, rozhraních, paměti a konfiguračním registru.
show clock	Příkaz zobrazí nastavený datum a čas.
show history	Příkaz zobrazí seznam naposledy použitých příkazů.
show interfaces [...]	Příkaz zobrazuje všechny informace o rozhraních.
telnet [...]	Telnet klient (pro připojení ke vzdáleným systémům).
disconnect ...	Odpojí navázanou relaci. Zobrazí seznam navázaných spojení.
show sessions	<ul style="list-style-type: none"><li>• <b>Ctrl+Shift+6</b>, pak <b>x</b> - "uspí" aktivní relaci</li></ul>
terminal history [...]	Příkaz nastaví velikost bufferu pro uchovávání zadaných příkazů.

### Privilegovaný mód (Privileged Mode)

Mód, ve kterém je možno provádět změnu konfigurace.

disable

Způsobí návrat do uživatelského módu (User Mode).

```
setup
```

Spustí konfigurační utilitu (postupně se formou jednoduchých otázek zeptá na základní konfiguraci).

```
clock set [...]
```

Příkaz nastaví čas.

```
debug ...
```

Zapne různé ladící výpisy.

```
debug ip rip
```

Zapne ladící výpisy pro routovací protokol RIP.

```
debug ip igrp ...
```

Zapne ladící výpisy pro routovací protokol IGRP.

```
debug all
```

Zapne všechny ladící výpisy.

```
debug ppp negotiation
```

Zapne ladící výpisy pro ověřování v protokolu PPP.

```
clear ...
```

Resetovací funkce.

```
clear arp-cache
```

Vymaže ARP cache (tabulka obsahující k IP adresám jejich MAC adresy).

```
clear ip route *
```

Vymaže routovací tabulku.

```
config  
configure terminal
```

Vstoupí do konfiguračního módu.

```
copy running-config tftp
```

Zkopíruje současnou konfiguraci na TFTP server (soubor).

```
reload
```

Restartuje IOS.

**Konfigurační mód (Configuration Mode)**

Hlubší konfigurace routeru/switche.

```
end
```

- nebo klávesová zkratka **Ctrl+Z** návrat do privilegovaného módu

```
hostname ...
```

Příkaz nastaví jméno zařízení (routeru/switche).

```
line ...
```

Nastavuje způsob přihlašování ke konzoli a virtuálním terminálům.

```
line console 0
password <heslo>
login
```

Konfiguruje konzoli (potřeba zadat heslo).

```
line vty 0 4
password <heslo>
login
```

Konfiguruje virtuální terminály 0 - 4 (potřeba zadat heslo).

```
enable password <heslo>
```

Zapne potřebu zadání hesla při přechodu do konfiguračního módu.

```
enable secret <heslo>
```

Heslo pro přechod do konfiguračního módu (bude uloženo zašifrované).

```
interface <typ> [<slot>/]<port>
```

Vstoupí do konfigurace rozhraní.

```
banner motd <znak>
```

Nastaví popis routeru ukončený daným znakem.

```
ip ...
```

Konfiguruje IP protokol.

```
ip host <jméno> <adresy>
```

Přidá IP adresy pod jméno daného zařízení.

```
boot system ...
```

Nastaví, z kterého umístění se má při startu natáhnout IOS.

## Konfigurace rozhraní (Interface Mode)

Příkazy společné nastavování různých rozhraní.

```
ip ...
```

Nastavuje protokol IP.

```
ip address <adresa> <maska>
```

Nastaví na daném rozhraní danou IP adresu.

```
shutdown
```

Vypne rozhraní.

```
no shutdown
```

Zapne rozhraní.

```
description <popis>
```

Nastaví popis daného rozhraní.

## **[\[editovat\]](#) Konfigurace routeru**

Tyto příkazy fungují pouze na směrovačích (ne na přepínačích).

### **Přístup přes SSH**

Postup při konfiguraci přístupu přes SSH

```
hostname <jmeno>
```

Konfigurace jména routeru.

```
ip domain-name <domena.com>
```

Konfigurace jména domény.

```
crypto key generate rsa
```

Vygenerování rsa klíče.

```
username <jmeno_uzivatele> secret <heslo>
```

Vytvoření uživatele s heslem

```
line vty 0 4  
transport input ssh  
login local
```

Vytvoření virtuálních terminálů 0 - 4 s přístupem přes ssh a pro přihlašování se použije lokální uživatel s heslem

### **CDP**

Práce s Cisco Discovery Protocol.

```
cdp run
```

V konfiguračním módu - zapne CDP na všech rozhraních.

```
cdp enable
```

V konfiguraci rozhraní - zapne CDP na daném rozhraní.

```
show cdp
```

Zobrazí informace o CDP.

```
show cdp entry ...
```

Zobrazí informace o daném sousedovi.

```
show cdp interface
```

Zobrazí informace o rozhraních, na kterých běží CDP.

```
show cdp neighbors
```

V privilegovaném módu - zobrazí sousedy.

```
clear cdp table
```

Smaže tabulku sousedů.

```
clear cdp counters
```

Resetuje čítače přenosů dat.

```
show cdp traffic
```

Zobrazí množství přenesených dat mezi sousedy.

```
debug cdp ...
```

Zapne ladící výpisy CDP.

```
cdp timer
```

Nastaví interval zasílání CDP updatů.

## ACL

Práce s ACL (Access Control List).

```
access-list <číslo typu protokolu> {deny|permit} <adresa> <wildchar maska>
```

V konfiguračním módu přidá pravidlo do seznamu s daným číslem (pro IP 1-99).

```
ip access-group <číslo acl> ...
```

V konfiguraci rozhraní aplikuje ACL (i pojmenovaný) na daný traffic (in/out).

```
show access-lists
```

Zobrazí seznam ACL.



```
access-list <číslo typu protokolu> {deny|permit} <protokol> <adresa> <wildchar mas-  
ka> ...
```

Rozšířené IP ACL (s čísly 100 - 199). Možnost filtrace podle transportního protokolu i podle čísel portů.

```
ip access-list extended <jméno>
```

Vytváří pojmenovací ACL.

### [\[editovat\]](#) Statické směrování

Nastavování statických routovacích záznamů.

```
ip route <cílová síť> <maska> <odchozí rozhraní>
```

V konfiguračním módu přidá statický směrovací záznam.

```
ip route 0.0.0.0 0.0.0.0 <odchozí rozhraní>
```

Výchozí brána pro nepřímo připojené sítě.

```
show ip route
```

Zobrazí směrovací tabulku.

```
ip default-network <adresa sítě>
```

Nastaví defaultní cestu.

### [\[editovat\]](#) RIP směrování

Nastavování routovacího protokolu RIP (Routing Information Protocol).

```
router rip
```

Vstoupí do konfigurace protokolu RIP (z konfiguračního módu).

```
network <síť>
```

Přidá síť, se kterou má protokol RIP pracovat.

```
no network <síť>
```

Odebere síť.

```
version <verze>
```

Nastaví verzi protokolu RIP (1/2).

```
show ip route
```

Zobrazí routovací tabulku.

```
show ip protocols
```

Zobrazí informace o routovacím protokolu.

```
debug ip rip
```

Zapne ladící výpisy.

```
show ip rip database
```

Zobrazí databázi cest protokolu RIP.

```
passive-interface <rozhraní>
```

Na daném rozhraní se bude pouze naslouchat (ne vysílat).

### [\[editovat\]](#) IGRP směrování

Nastavování routovacího protokolu IGRP (Interior Gateway Routing Protocol).

```
router igrp ...
```

Vstoupí do režimu konfigurace protokolu IGRP. Parametrem je číslo autonomního systému.

```
network <síť>
```

Přidá síť, se kterou má IGRP pracovat.

```
no network <síť>
```

Odebere síť.

```
show ip route
```

Zobrazí routovací tabulku.

```
show ip protocols
```

Zobrazí informace o IGRP (intervaly updatů, ...).

```
no router igrp ...
```

Vypne IGRP.

```
debug ip igrp ...
```

Zapne ladící výpisy.

### [\[editovat\]](#) OSPF směrování

Konfigurace OSPF (Open Shortest Path First).

```
router ospf <číslo procesu>
```

Vstoupí do konfigurace OSPF.

```
network <adresa> <wildchar maska> area <area ID>
```

Přidá síť, area ID pro pátevní síť musí být rovno 0.

```
area <area ID> authentication message-digest
```

Zapne autentizaci mezi routery.

```
ip ospf priority <priorita>
```

V konfiguraci rozhraní - nastaví prioritu routeru (0 - 255), vyšší priorita = větší pravděpodobnost zvolení routeru jako DR.

```
ip ospf cost <číslo ceny cesty>
```

Nastaví cenu cesty - vyšší číslo = horší cesta.

```
ip ospf message-digest-key 1 md5 7 <heslo>
```

Nastaví klíč (heslo) pro autentizaci mezi routery.

```
ip ospf hello-interval <čas>
```

Nastaví interval mezi hello pakety.

```
ip ospf dead-interval <čas>
```

Interval, kdy je router považován za ztracený (pokud se neozve).

```
show ip route
```

Zobrazí routovací tabulku.

```
show ip ospf [...]
```

Zobrazí různé informace o OSPF.

```
show ip ospf database
```

Zobrazí databázi cest.

```
debug ip ospf [...]
```

Zapne ladící výpisy (např. events).

## [\[editovat\]](#) EIGRP směrování

Nastavení směrovacího protokolu EIGRP (Enhanced Interior Gateway Routing Protocol).

```
router eigrp <číslo autonomního systému>
```

V konfiguračním módu. Vstoupí do konfigurace EIGRP. Číslo autonomního systému může korespondovat s IGRP (EIGRP přeposílá cesty IGRP se stejným číslem).

```
network <síť>
```

Přidá síť, se kterou má EIGRP pracovat.

```
show ip route
```

Zobrazí routovací tabulku.

```
show ip eigrp neighbors
```

Zobrazí sousední routery s EIGRP.

```
show ip eigrp topology
```

Zobrazí tabulku topologie sítě.

```
debug eigrp ...
```

Zapne různé ladící výpisy (např. packet).

```
auto-summary
```

V konfiguraci routování. Zapne autosumarizaci (shlukování záznamů s různými maskami podsítí).

```
ip summary-address eigrp <autonomní číslo> <sítě> <maska>
```

Autosumarizace na rozhraní.

```
ip authentication mode eigrp <číslo aut. systému> md5  
ip authentication key-chain eigrp <aut. č.> <název klíče>
```

V konfiguraci rozhraní. Zapne autentizaci MD5 klíčem.

```
key chain <název klíče>
```

V konfiguračním módu. Vstoupí do konfigurace klíče.

```
key <číslo klíče>  
key-string <text>
```

Nastaví klíč.

## NAT/PAT

Konfigurace překladu adres NAT/PAT (Network Address Translation/Port Address Translation).

```
ip nat inside source static <vnitřní IP> <vnější ip>
```

Provede pevnou (statickou) konfiguraci NAT (vnitřní IP <=> vnější IP). V konfiguračním režimu.

```
ip nat inside
```

V konfiguraci rozhraní. Identifikuje rozhraní na vnitřní síti.

```
ip nat outside
```

V konfiguraci rozhraní. Identifikuje rozhraní na vnější síti.

```
ip nat pool <jméno> <od IP> <do IP> netmask <maska sítě>
```

Vytvoří pojmenovaný rozsah IP adres pro dynamický NAT.

```
ip nat inside source list <číslo ACL> pool <jméno>
```

Definuje dynamický nat s daným ACL a daným rozsahem adres. ACL by měl povolovat vnitřní rozsah (permit).

```
ip nat inside source list <číslo ACL> interface <rozhraní> overload
```

Definuje PAT (využívá čísla portů k adresování počítačů ve vnitřní síti).

```
clear ip nat translation *
```

Smaže všechny dynamické záznamy.

```
show ip nat translation
```

Zobrazí aktivní překlady.

## **DHCP**

Konfigurace DHCP (Dynamic Host Configuration Protocol).

```
ip dhcp pool <jméno>
```

Vstoupí do konfigurace DHCP.

```
network <adresa> <maska>
```

Specifikuje rozsah přidělovaných adres.

```
default-router
```

Specifikuje přidělovanou výchozí bránu.

```
dns-server
```

Přidělovaný DNS server.

```
ip dhcp excluded-address <od IP> <do IP>
```

Adresy, které nebudou přidělovány. V konfiguračním módu.

```
show ip dhcp binding
```

Zobrazí tabulku propůjčených adres.

```
debug ip DHCP server events
```

Zapne ladící výpisy.

```
ip helper-address <adresa>
```

V konfiguraci rozhraní. Vytvoří DHCP relay. Router se bude tvářit jako DHCP a bude komunikovat se skutečným DHCP serverem.

## **PPP**

Nastavení protokolu PPP (Point-to-Point Protocol).

```
encapsulation ppp
```

Při konfiguraci rozhraní. Zapne PPP.

```
ppp authentication [PAP|CHAP]
```

Zapne algoritmus autentizace.

```
ppp pap sent-username <jméno> password <heslo>
```

Jméno, kterým se identifikuje rozhraní.

```
username <jméno> password <heslo>
```

V konfiguračním módu. Přidá vzdáleného uživatele.

```
compress [predictor|stac]
```

Nastaví algoritmus komprese.

```
debug ppp authentication
```

Zapne ladící výpisy autentizace PPP.

## ISDN

Konfigurace ISDN (Integrated Services Digital network).

```
isdn switch-type <typ switche>
```

Nastavuje typ ISDN switche, se kterým router komunikuje. Např. basic-ni.

```
isdn spid{1|2} <SPID číslo> [<lokální číslo>]
```

Nastaví přiřazené SPID číslo. V konfiguraci rozhraní BRI.

```
controller <controller> <slot>/<port>
```

Vstoupí do menu controller.

```
pri-group timeslots <rozsah>
```

Nastaví port jako PRI.

```
show isdn status
```

Zobrazí informace o ISDN spojení.

```
show interface bri0/0
```

Zobrazí statistiky pro konkrétní rozhraní routeru.

```
show isdn active
```

Zobrazí probíhající spojení.

```
debug isdn ...
```

Ladící výpisy.

```
dialer-list <číslo> protocol op permit
```



Vytvoří vytáček seznam.

```
dialer-group <číslo>
```

Přiřadí rozhraní do vytáček seznamu. V konfiguraci rozhraní BRI.

### [\[editovat\]](#) **Frame-Relay**

Nastavování Frame relay.

```
encapsulation frame-relay [IETF]
```

Zapne na daném rozhraní frame-relay. V konfiguraci rozhraní.

```
bandwidth <číslo>
```

Nastaví šířku přenosu. např. 64.

```
frame-relay lmi-type ansi
```

Nastaví se jen na jedné straně linky.

```
frame-relay map ip <adresa druhé strany> <číslo> broadcast [IETF]
```

Nastaví routeru adresu druhé strany spoje.

```
show frame-relay map
```

Zobrazí tabulku frame-relay routerů.

```
debug frame-relay ...
```

Ladící výpisy.

### [\[editovat\]](#) **Konfigurace switche**

Příkazy pouze pro přepínače (ne směrovače).

### [\[editovat\]](#) **Layer 3 přístup**

Konfigurace L3 přístupu k přepínači.

```
Switch(config)#interface VLAN 1
Switch(config-if)#ip address 192.168.1.2 255.255.255.0
Switch(config-if)#exit
Switch(config)#ip default-gateway 192.168.1.1
Switch#show interace VLAN 1
```

Nastaví IP adresu na VLAN switche.

### [\[editovat\]](#) **STP**

Konfigurace STP (Spanning Tree Protocol).

```
show spanning-tree brief
```

Zobrazí zprávu o STP.

```
spanning-tree vlan 1 priority <priorita>
```

Nastaví prioritu STP na vlan 1.

### [\[editovat\]](#) **Správa tabulky MAC adres**

Konfigurace tabulky MAC adres.

```
show mac-address-table
```

Zobrazení tabulky.

```
show mac-address-table static
```

Zobrazení statických záznamů.

```
clear mac-address-table dynamic
```

Vymazání dynamických záznamu.

```
mac-address-table static <MAC adresa> interface <rozhraní> ...
```

Přidá statický záznam o MAC adrese do tabulky MAC adres. Možno specifikovat VLAN.

```
switchport port-security ...
```

Na daném rozhraní nastaví zabezpečení (např. jen pro danou staticky nastavenou MAC adresu).

### [\[editovat\]](#) **VLAN**

Konfigurace VLAN.

```
show vlan
```

Zobrazí tabulku VLAN.

```
vlan database
```

Vstoupí do konfigurace VLAN.

```
vlan <číslo> name <jméno>
```

Přidá VLAN.

```
no vlan <číslo>
```

Odebere VLAN.

```
delete flash:vlan.dat
```

Smaže databázi VLAN v paměti FLASH.

```
switchport mode {access|trunk}
```

V konfiguraci rozhraní. Nastaví mód přenosu přes spoj.

```
switchport access vlan <číslo>
```

Přiřadí rozhraní do VLAN.

```
no switchport access vlan <číslo>
```

Odebere rozhraní z VLAN.

```
switchport trunk encapsulation <typ>
```

Nastaví obalování paketů v režimu trunk na <typ>.

```
interface <rozhraní> <slot>/<port>.<subport>
```