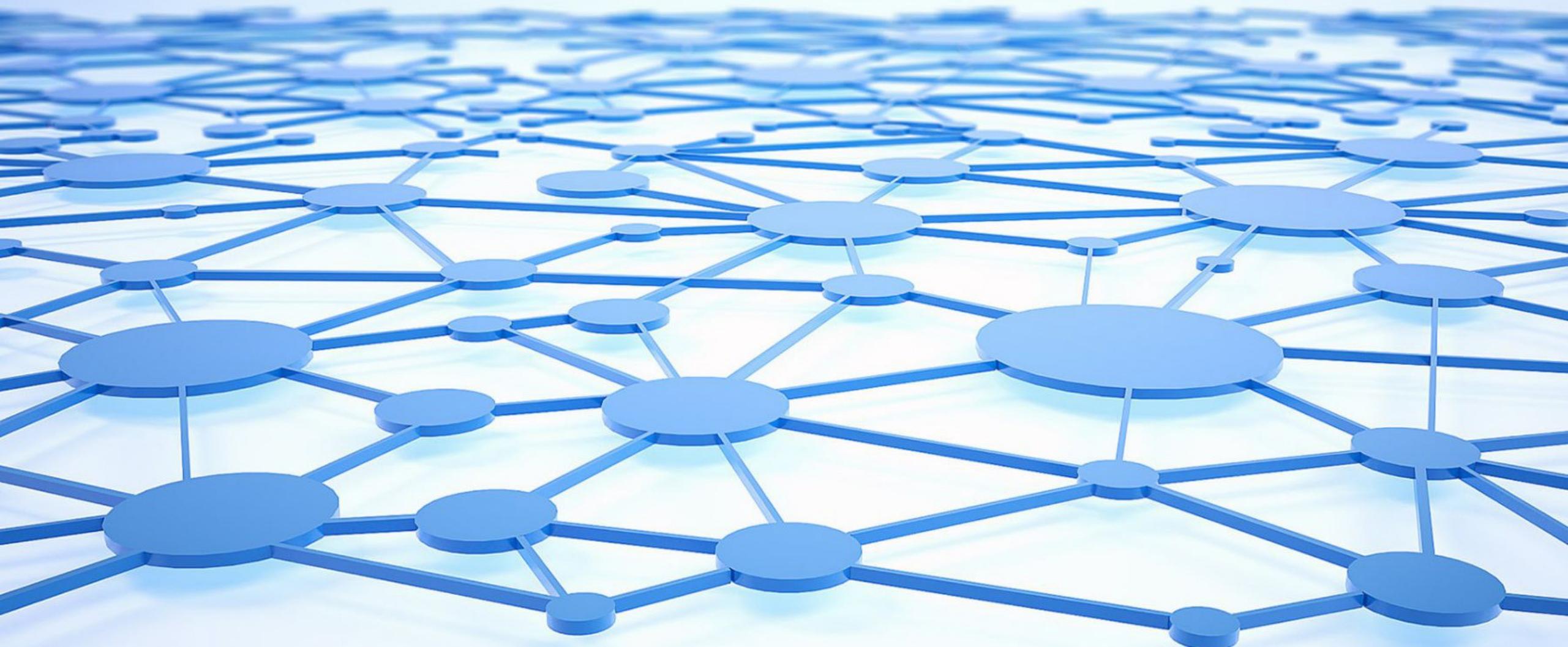


# Úvod do počítačových sítí

Přednáška 1 ( 2025/2026 )

ver. 2025-09-17-01



# Vyučující

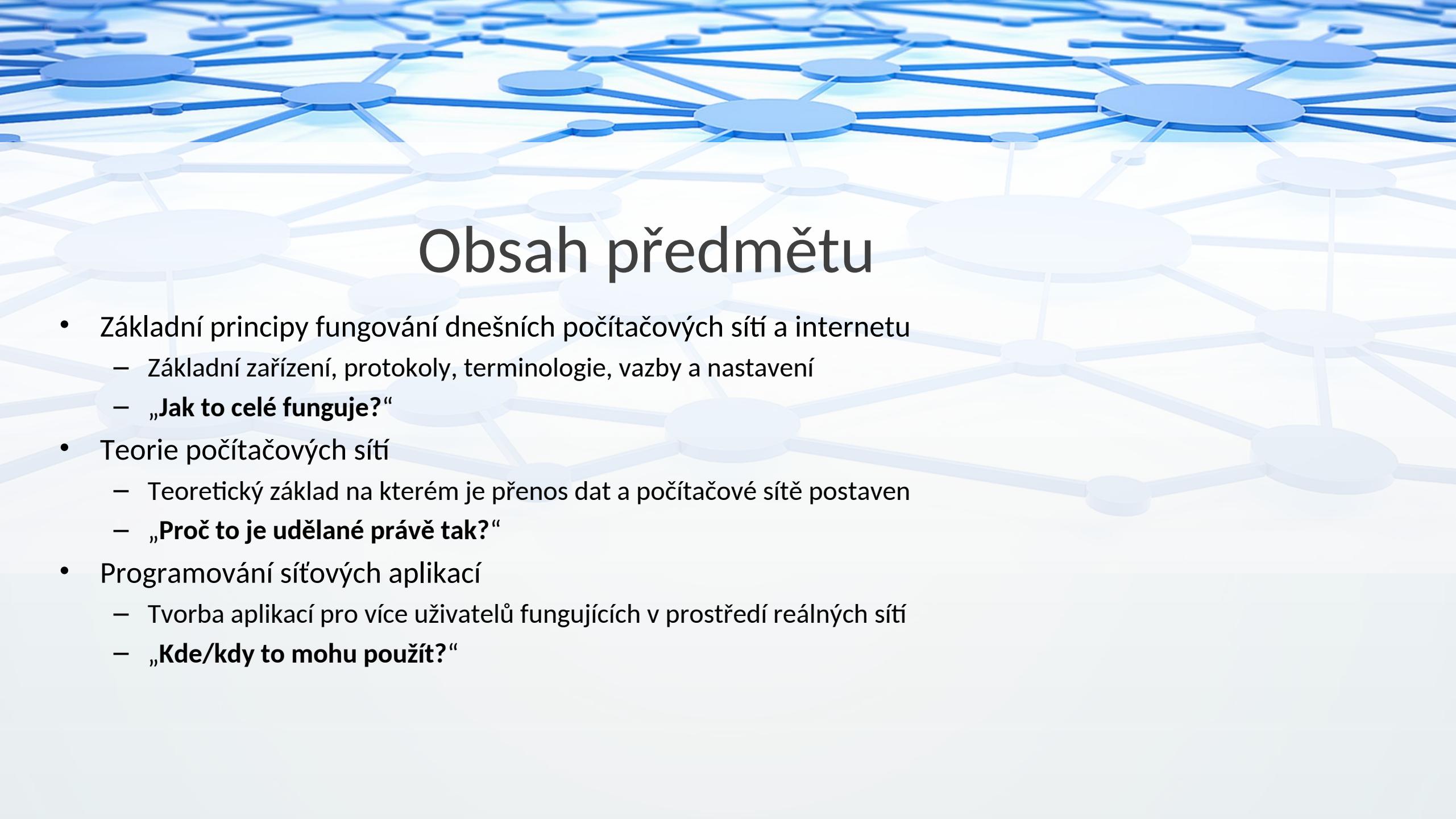
- Přednášky
  - Ing. Luboš Matějka, Ph.D., UN 358, lmatejka@kiv.zcu.cz
    - Konzultace St 12:00-12:45 – po předchozí domluvě nebo **emailem**
- Cvičení
  - Ing. Martin Úbl, UN305, ublm@kiv.zcu.cz
    - Konzultace Po 14:00-15:00; Čt 15:00-16:00
  - Ing. Jindřich Skupa, UN305, skupaj@kiv.zcu.cz
    - Konzultace **emailem** / po cvičení

# Podmínky získání zápočtu

- Získání alespoň 10 bodů v zápočtovém testu z 20 možných
  - Písemná forma ověření znalostí
  - 1 řádný termín ( **cca v týdnu 24.11. - 28.11.2025** ) a jeden opravný ( cca v týdnu 8 - 12.12.2025 )
    - Ještě upřesníme jak budeme daleko v cvičeních
- Získání alespoň 15 bodů ze semestrální práce z 30 možných
  - Bude se jednat o síťovou počítačovou hru pro více hráčů
    - Server v C/C++, klienti v Javě či jiném - mimo C/C++ a cvičícím schváleném jazyce
  - Návrh zvolené hry schvaluje Martin Úbl do 3 cvičení, **student sám přijde s návrhem - pošle emailem**
  - K 30 základním bodům je možné získat ještě 10 bonusových za mimořádně povedené zpracování
  - Mezní termín řádného odevzdání **15.1.2026**
    - Za každý den po řádném termínu se odečte 1 bod
  - Mezní termín získání zápočtu **15.2.2026**

# Podmínky zkoušky

- Získání zápočtu
- Získání alespoň **25 bodů z celkových 50 možných** z písemného testu
- Hodnocení je součtem bodů z
  - Zápočtového testu, samostatné práce, zkouškového testu
- Hodnocení
  - 85 a více bodů výborně
  - 70 až 84 bodů velmi dobře
  - 50 až 69 bodů dobré
  - 49 bodů a méně nevyhověl

A complex network diagram composed of numerous blue circular nodes of varying sizes and thin blue lines representing connections between them, creating a sense of a vast, interconnected system.

# Obsah předmětu

- Základní principy fungování dnešních počítačových sítí a internetu
  - Základní zařízení, protokoly, terminologie, vazby a nastavení
  - „**Jak to celé funguje?**“
- Teorie počítačových sítí
  - Teoretický základ na kterém je přenos dat a počítačové sítě postaven
  - „**Proč to je udělané právě tak?**“
- Programování síťových aplikací
  - Tvorba aplikací pro více uživatelů fungujících v prostředí reálných sítí
  - „**Kde/kdy to mohu použít?**“

# Rámcový plán přenášek I.

1. Informace o předmětu, co je počítačová síť, dělení sítí dle velikosti, základní prvky v sítě/Internetu( switch, hub, router, firewall, modem )
2. MAC adresy, IPv4, třídy adres, masky sítí, NAT, základní protokoly v internetu( DHCP, ARP, DNS, ICMP )
3. ISO/OSI model, TCP/IP model, zapouzdřování protokolů. Fyzická vrstva: základní funkce, zařízení fyzické vrstvy, topologie zapojení, typy spojů a přenosů, digitální a analogový přenos, Fourierova analýza, modulační a přenosová rychlosť, Nyquistovo a Shannonovo kritérium, asynchronní, arytmický a synchronní přenos, využití kapacity přenosového kanálu, kódování signálu ( RZ, RZI, NRZ, NRZI, Manchester, Dif. Manchester )
4. Fyzická vrstva: Přenos v základním a přeneseném pásmu, modulace ( frekvenční, amplitudová, fázová ), multiplex ( časový, vlnový, frekvenční ), zpoždění, latence, RTT, přenosová media
5. Linková vrstva a její dělení ( LLC, MAC ), bitově a znakově orientované přenosy, zajištění transparentnosti přenosu, detekce chyb ( parita, sumy, CRC )
6. Linková vrstva - AQK, FEC, Hammingova vzdálenost, samoopravné kody, Stop&Wait, číslování rámců, samostatné a kontinuální potvrzování, protokoly s klouzajícím okénkem, Go back N, Selective repeat, řízení toku dat
7. Linková vrstva – metody řízení přístupu k přenosovému médiu ( Aloha, CSMA, CSMA/CD ), pověření a prioritizace, příklady protokolů linkové vrstvy ( Ethernet, Token Ring ), transportní mosty, vlany

# Rámcový plán přenášek II.

8. Síťová vrstva – store & forward, směrovací a forwardovací tabulka, typy směrování, statické a dynamické směrování ( LVA, DVA ), IGP, EGP, autonomní systémy
9. Síťová vrstva – Unicat, Broadcast, Multicast
10. Transportní vrstva – spojovaný a nespojovaný přenos ( UDP, TCP ), QoS
11. Relační, prezentační a aplikační vrstva, Nejpoužívanější aplikační protokoly v Internetu ( HTTP, SMTP, POP3/IMAP, SNMP )

# Rámcový plán cvičení I.

1. Základní orientace v Linuxu a nastavení sítě v Linuxu
2. Základní nastavení sítě ve Windows, debugovací nástroje pro trasování, ARP, DNS, sniffer
3. Programování síťových aplikací v C
  - **Mezní termín výběru zadání semestrální práce**
4. Programování síťových aplikací v Java
5. Programování síťových aplikací v heterogenním prostředí ( Linux versus Windows, C versus Java )
6. Principy návrhu komunikačního protokolu
7. Procvičování příkladů z přednášek
8. Procvičování příkladů z přednášek
9. Procvičování příkladů z přednášek
10. **Zápočtový test**

# Rámcový plán cvičení II.

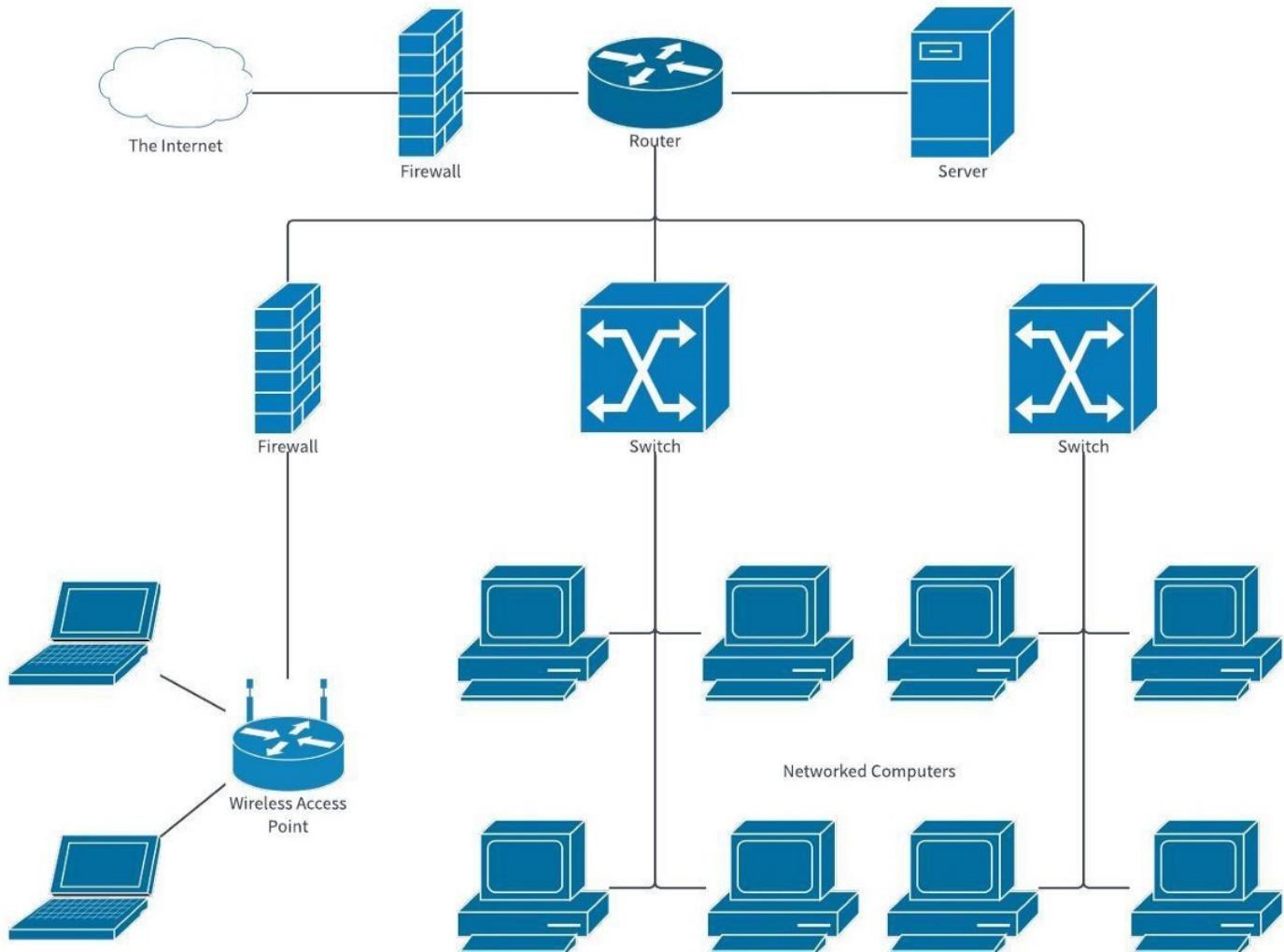
11. Základní konfigurace L2 ve virtualizaci ( STP, Vlan )
12. Základní konfigurace L3 ve virtualizaci ( IP, dynamický routing )
13. Konzultace semestrálních prací a první možný termín odevzdání semestrální práce

**!!! POZOR – docházka na cvičeních se bude evidovat !!!**

# Co je počítačová síť ?

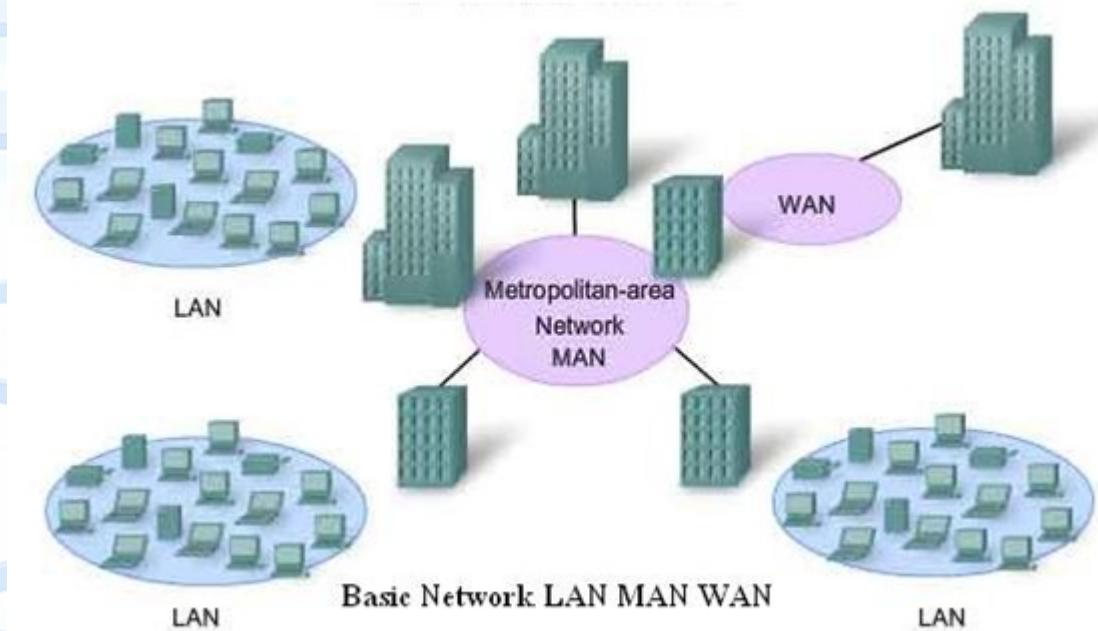
- Počítačová síť je soubor zařízení propojených komunikační sítí, dovolujících sdílet prostředky jako jsou data, programy nebo periferie, ....
- Počítačovou síť můžeme popisovat jako graf
  - To je důležité, protože mnoho úloh v počítačové síti se řeší jako grafové úlohy
  - Uzly - „to co má smysl propojovat“
    - Koncové uzly jako PC, notebooky, servery, mobilní telefony,....
    - Síťové prvky jako opakovače, huby, switche, routery
  - Hrany - „čím/kudy přenášíme data“
    - Komunikační medium
      - Cesta / přenosové cesta
        - Zajišťuje přenos signálů, například různé druhy kabelů
      - Kanál / přenosový kanál
        - Souhrn prostředků zajišťujících telekomunikační spojení dvou míst
        - Kanál uvažujeme typicky jednosměrný
      - Okruh
        - Obousměrný přenosový kanál

# Příklad počítačové sítě



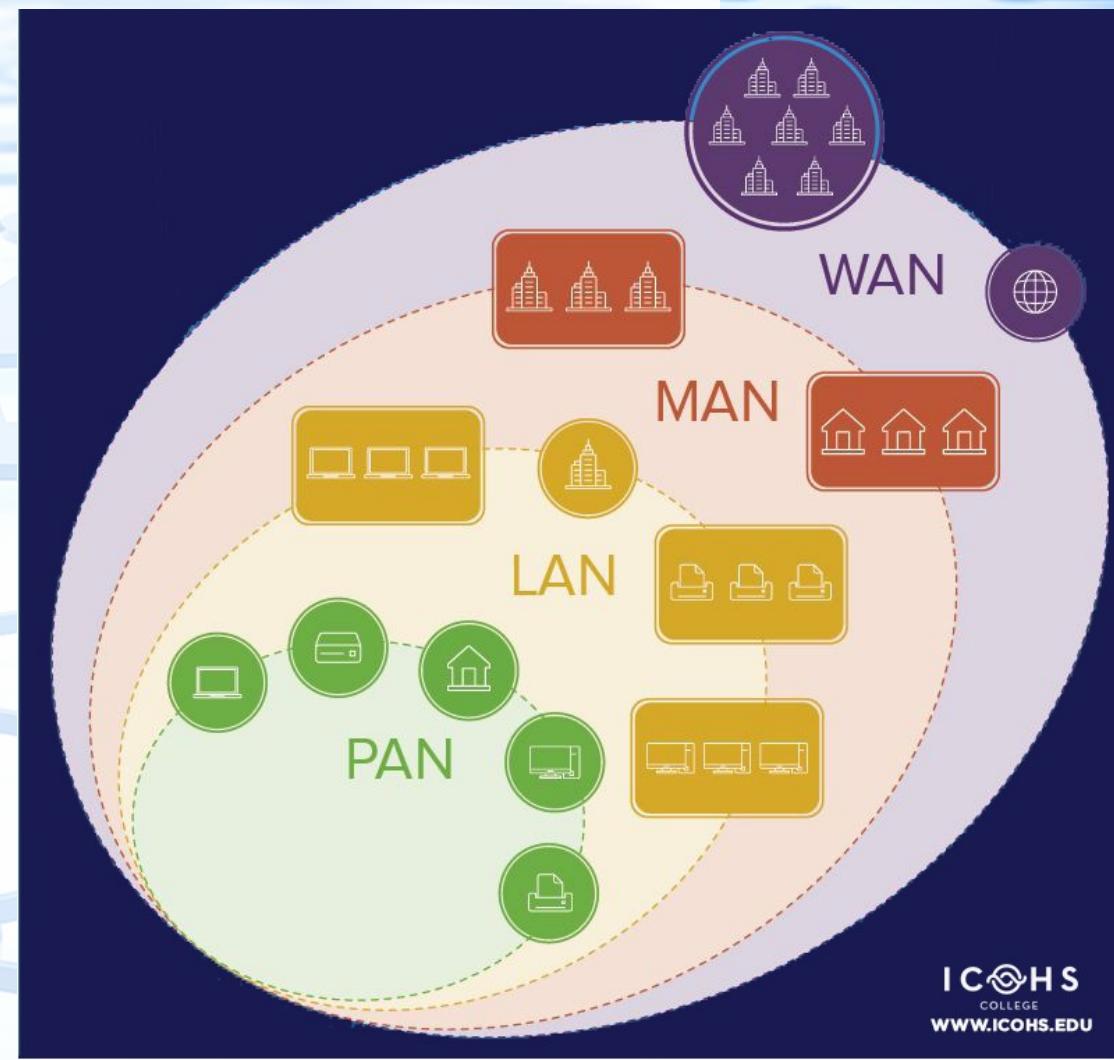
# Segmentace počítačových sítí

- Není možné řešit celý svět jako jednu velkou centralizovanou síť
  - Technicky téměř nemožné
  - Výpadek jednoho prvku by mohl znepřístupnit celý systém
- Nutnost rozdělení sítí do menších částí s lokálními pravidly a technologiemi
- Vzájemná interakce sítí jen na definovaných bodech
  - Různé sítě mohou používat různé technologie
  - Na propojovacím bodě ( směrovači ) si ale MUSEJÍ rozumět
  - Uvnitř sítě fungují jen lokální pravidla a protokoly



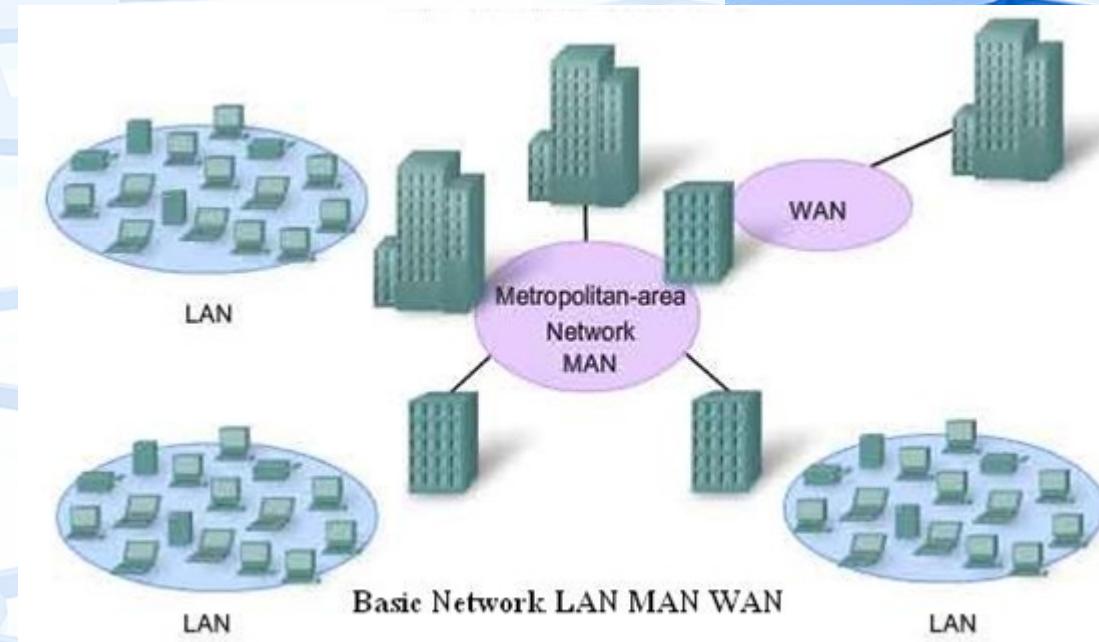
# Dělení sítí dle velikosti

- PAN
  - Personální, ~10m ( kolem osoby )
- LAN
  - Lokální, ~1km ( pokoj, budova, firma )
- MAN
  - Metropolitní, ~10km ( město )
- WAN
  - Rozsáhlé, ~100 ... 1000 ... km ( stát )



# Propojení a význam jednotlivých typů sítí

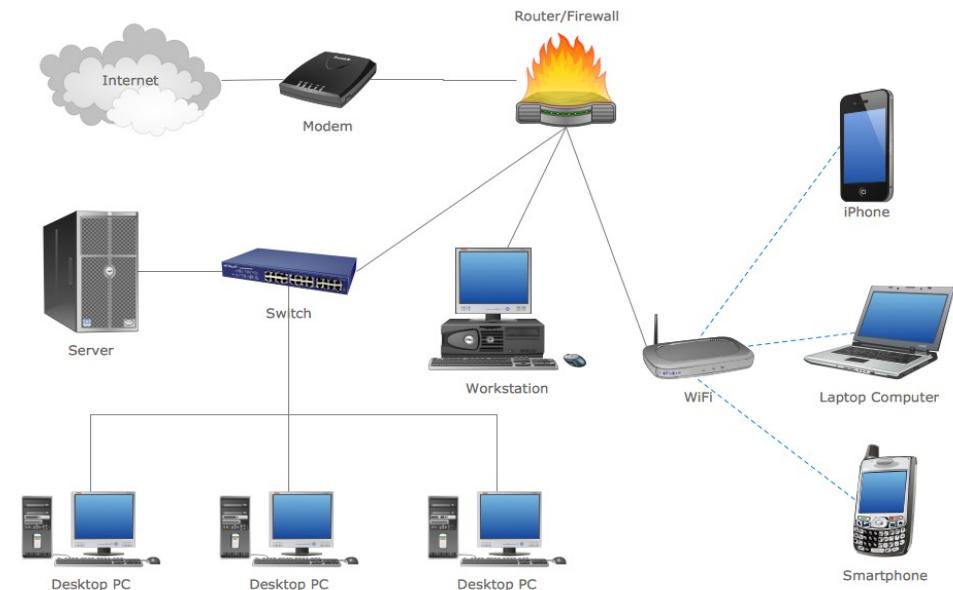
- Jednotlivé typy sítí se hierarchicky propojují a vytvářejí větší celky – například Internet
- Každá síť má primárně jiný účel a jiná pravidla
  - PAN – personální síť
    - Malé množství zařízení komunikujících na velice krátkou vzdálenost
    - Například pomocí bluetooth technologie
  - LAN – lokální připojení koncových zařízení
    - PC, notebooky, servery
    - V různých LAN mohou být použity různé technologie
  - MAN
    - Propojení více LAN mezi sebou
    - Směrování, větší datové toky než v LAN
  - WAN
    - Propojení více LAN/MAN/WAN prostřednictvím ISP
      - ISP – Internet Service Provider
    - Výrazně menší počet zařízení a menší počet změn než v LAN
    - Vyšší požadavky na kapacitu

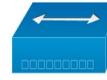


# Typické uzly počítačových sítí typu Internet

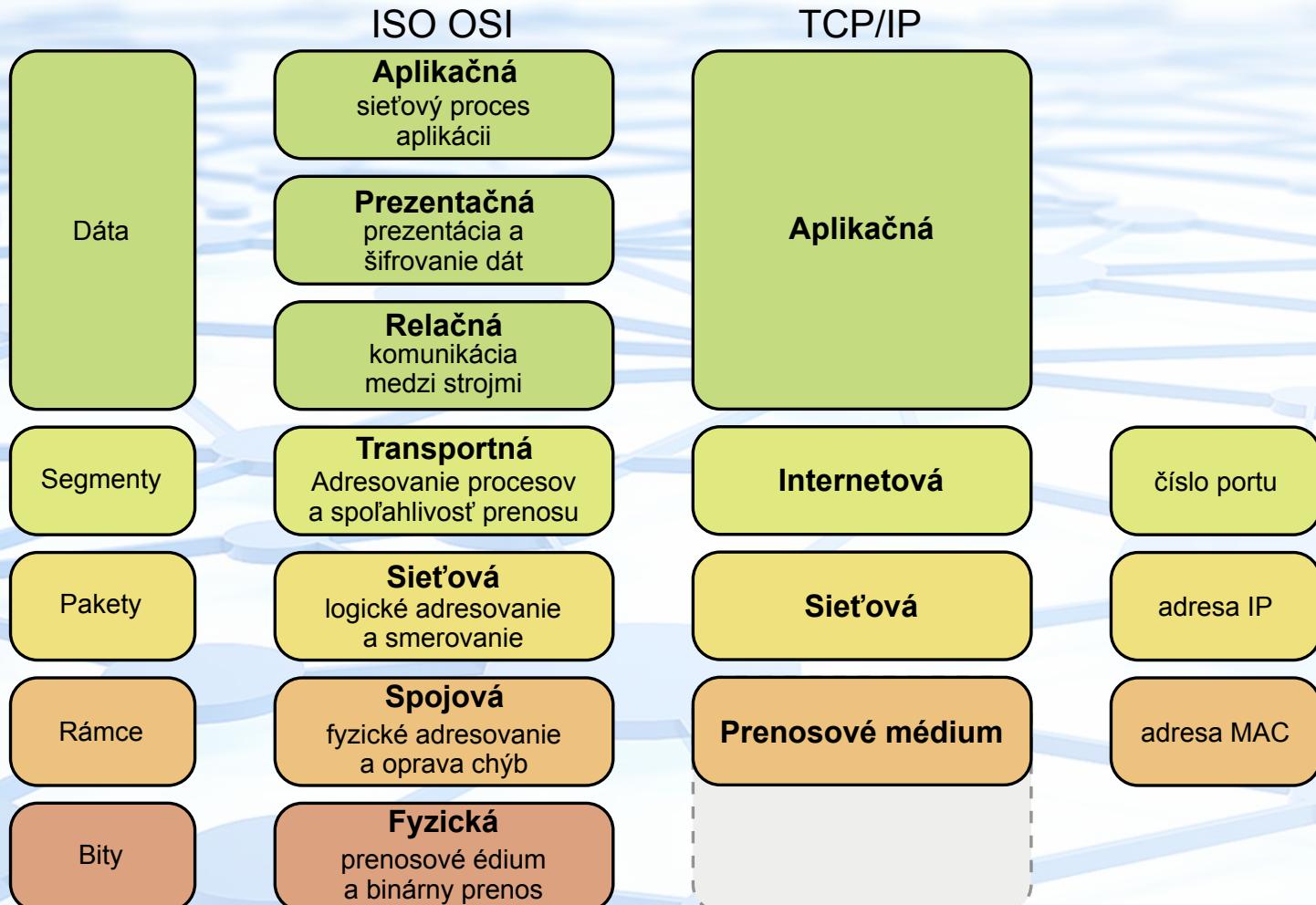
- PC, servery, notebooky, mobilní telefony, chytré televize, ledničky, automobily ....
  - Typicky se jedná o koncové uzly – poskytují nebo požadují data
  - Ve výjimečných případech mohou data jen přenášet – Linux jako LAN-WAN router
- Opakovač / zesilovač signálu
  - Zesiluje signál na „dlouhých“ spojích, kde dochází k útlumu
- Switch a hub
- Router
- Modem
- Firewall

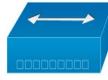
Network Diagram



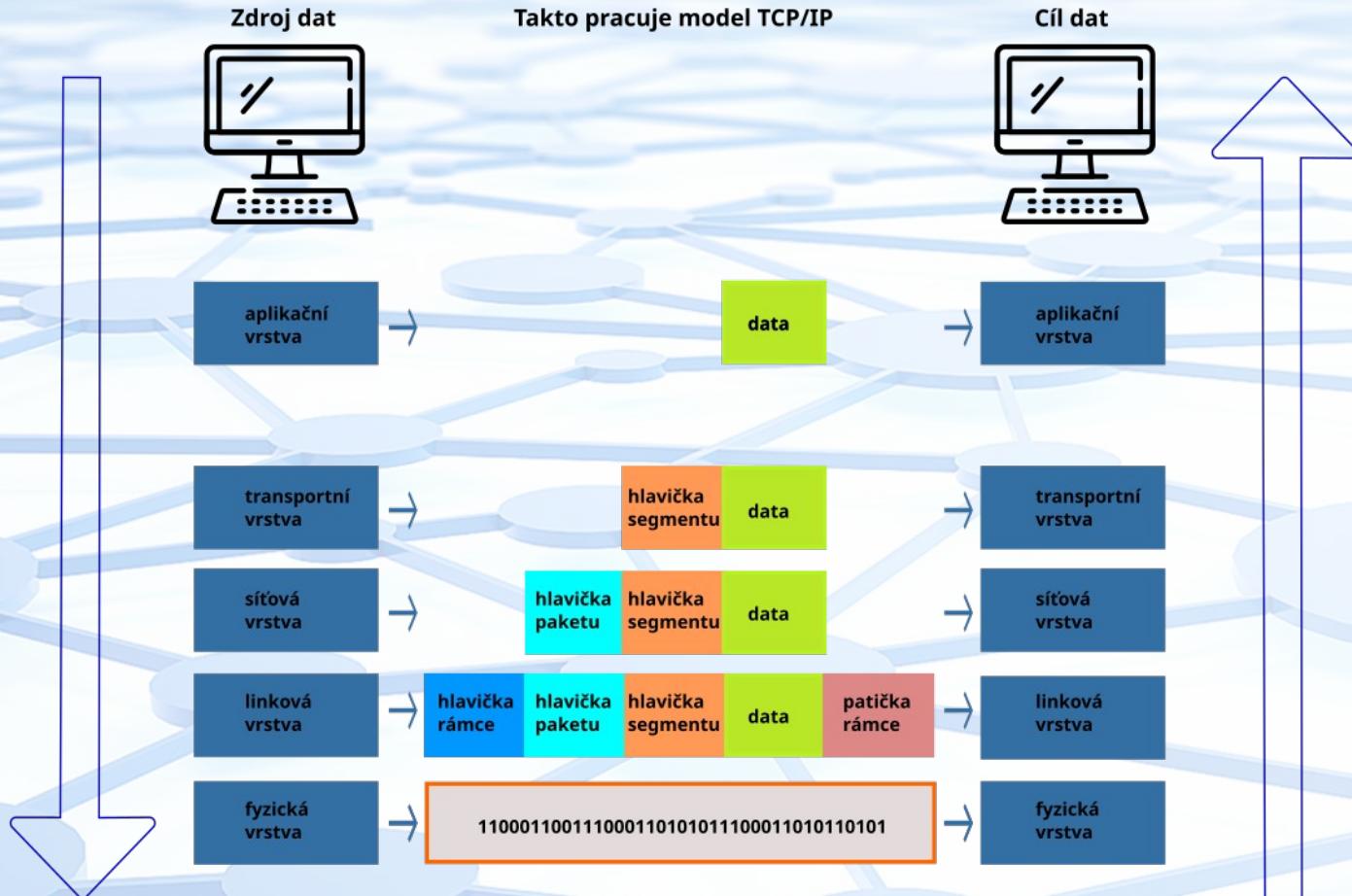


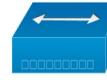
# Modely pro popis sítí





# Vztahy přenášených dat





# Aktivní síťové prvky - Hub

- Předchůdce switchů
- Jedná se defakto o více-portový opakovač
- Přijme signál a zopakuje jej na všechny porty kromě portu ze kterého přišel
- Nevidí rámce, ale pouze opakuje signál
  - Jinak řečeno „nerozumí přenášeným datům“
- Velký problém s bezpečností neboť všechn provoz na hubu vidí všichni připojení
- Chová se jako „jeden společný komunikační kanál“
- Může docházet ke kolizím
  - Nižší propustnost než switch – typicky 10/100 Mbps
- Dnes se téměř nepoužívá



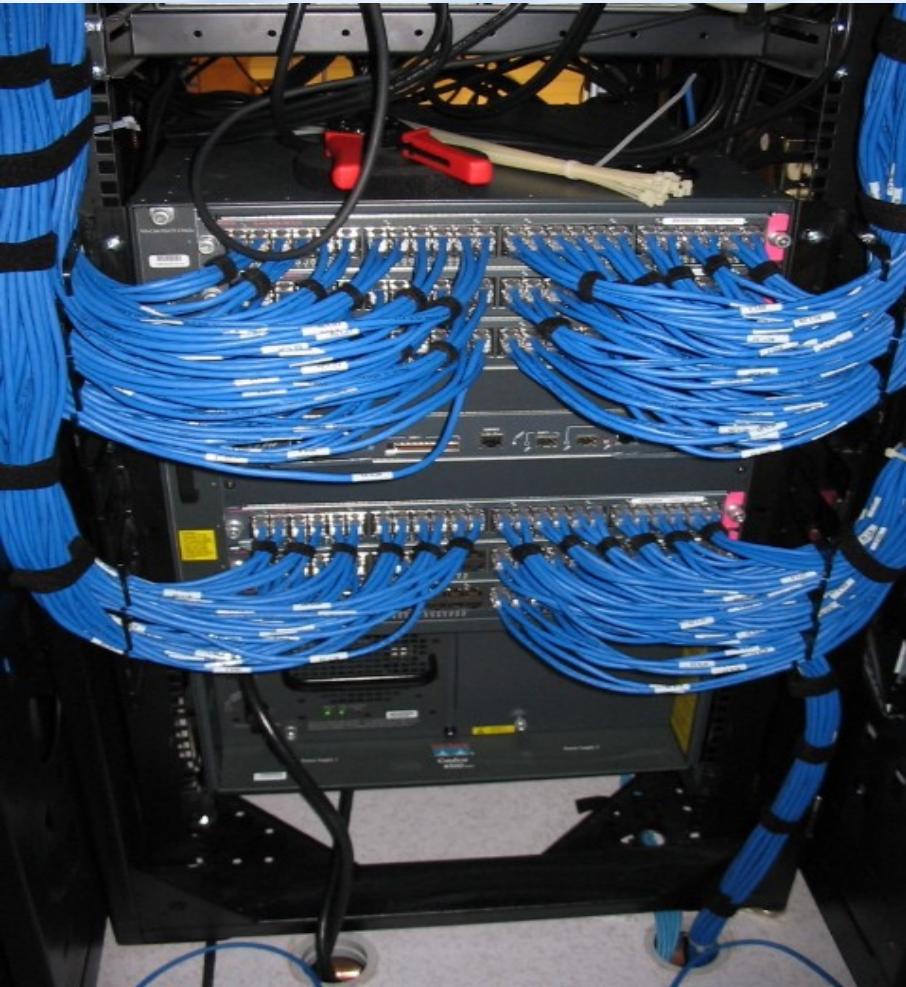
# Aktivní síťové prvky - Switch

- Spojuje síťový provoz z více uzlů do jednoho bodu
  - Umožňuje komunikaci uzlů navzájem
- Téměř vždy jednorázová HW zařízení
  - Na rozdíl od routeru/směrovače – viz dále
- Typicky více portů stejného typu
- Switche mohou podporovat různé rychlosti např. 10/100/1.000/10.000/100.000 Mbps
- Příchozí běžný rámec zkopíruje na port u kterého má v tabulce adres adresu příjemce
- Speciální příklad je broadcastový rámec – ten je zkopirován na všechny porty kromě portu příchozího
  - Pokud vznikne kruh, může vzniknout broadcastová bouře – data se neustále kopírují dokola
    - „Hloupější“ switche si s tím neporadí a po čase přestanou stíhat předávat provoz
    - „Chytřejší“ switche používají STP protokol, který hledá v síti smyčky a ty softwarově rozpojuje
    - Bouře se šíří v rámci broadcastové domény přes všechny switche
    - Šíření bouře limituje router/směrovač

# Aktivní síťové prvky – Switch -příklad I.



# Aktivní síťové prvky – Switch -příklad II.

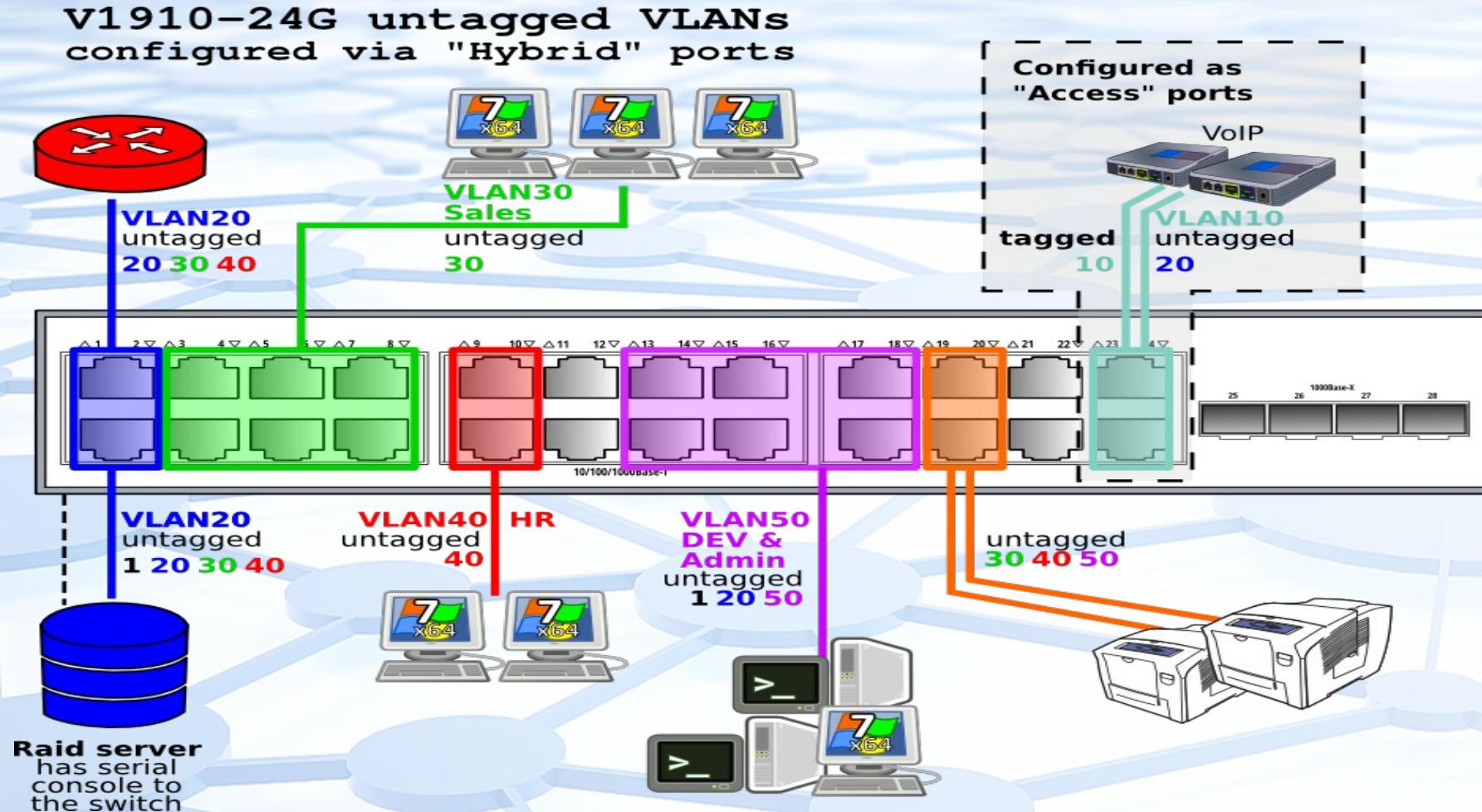


# Aktivní síťové prvky – Switch – VLAN, Trunk



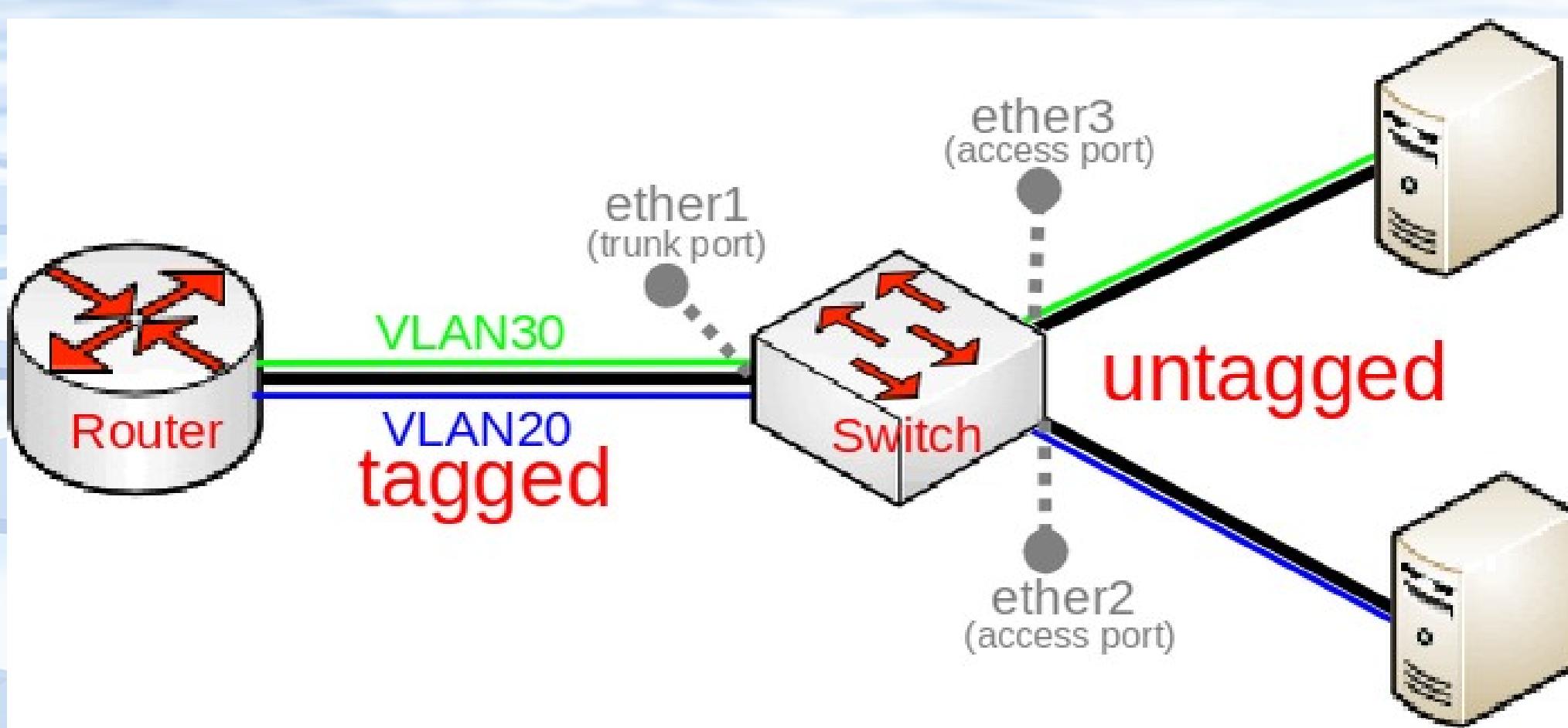
- VLAN virtuální LAN
  - „rozdělení“ jednoho fyzického zařízení na více virtuálních
  - Provoz v každé VLAN je isolované == nevidí data jiné VLAN
  - Identifikace VLAN pomocí VLAN ID, celé kladné číslo, výchozí je 1
- Access port
  - Porty v jedné VLAN se označují jako „access“ - „accessové“ porty
  - V rámci jedná VLAN není nutné VLAN ID uvádět == „NETAGOVANÝ“ provoz
- Trunk port
  - Tím že se VLANy nevidí mezi sebou, musela by pro každou VLAN být samostatná cesta k dalšímu zařízení – switch / router
  - To je problém, protože pro 20 VLAN by se obsadilo 20 portů switche
  - Trunk je speciální port, kterým může procházet provoz více VLAN
  - Jednotlivé rámce se rozeznají pomocí VLAN ID v hlavičce rámce == „TAGOVANÝ“ provoz
  - Porty kde je povolen trunk jsou označované jako „trunkové“ porty

# Aktivní síťové prvky – Switch – VLAN - příklad



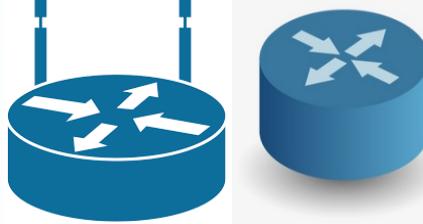
zdroj: <https://superuser.com/questions/704759/vlans-setup-via-untagged-hybrid-ports-on-procurve-switch>

# Aktivní síťové prvky – Switch – VLAN, Trunk, Access - příklad



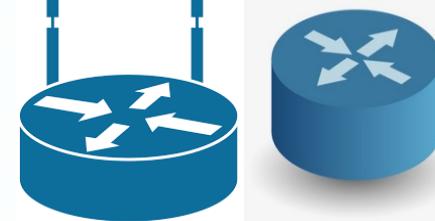
zdroj: [https://wiki.mikrotik.com/wiki/Manual:Bridge\\_VLAN\\_Table](https://wiki.mikrotik.com/wiki/Manual:Bridge_VLAN_Table)

# Aktivní síťové prvky - Router

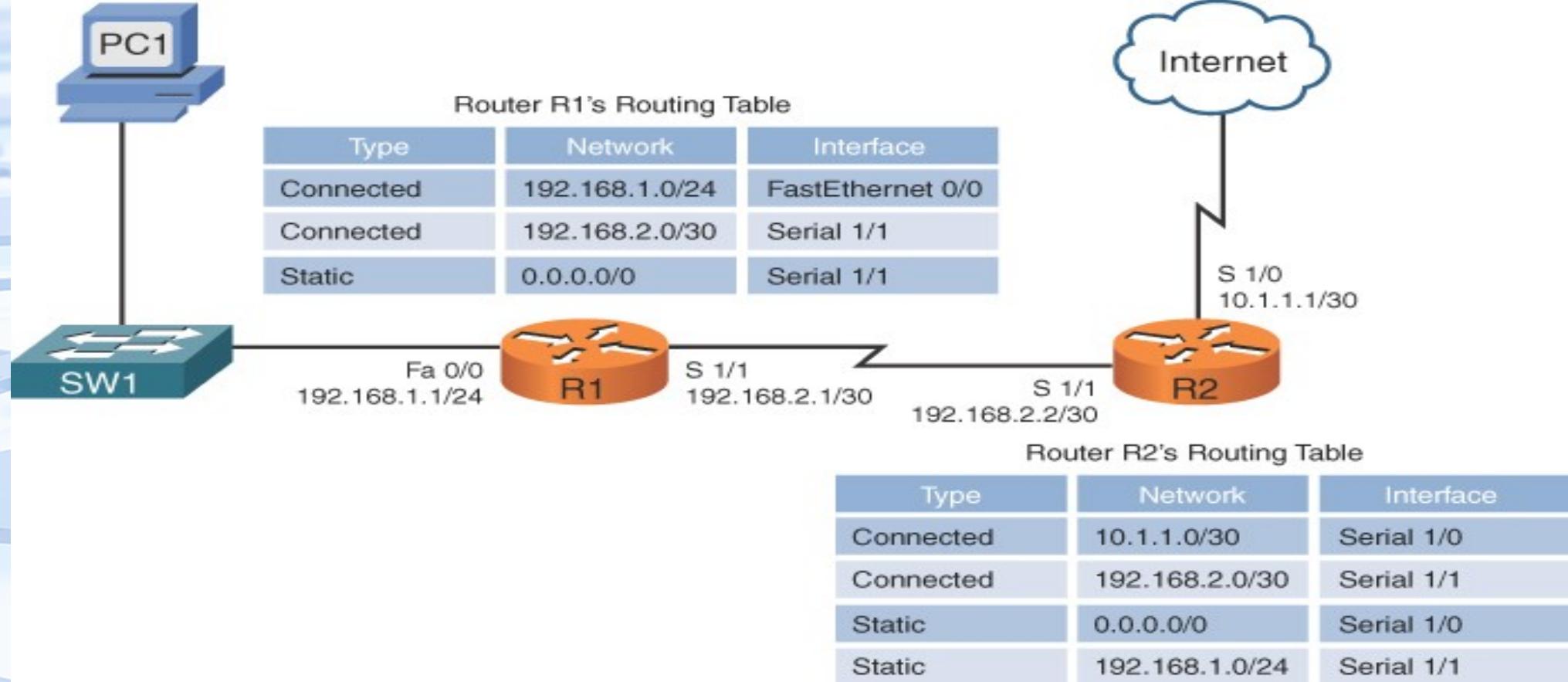
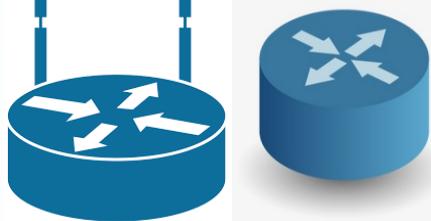


- Směruje provoz na úrovni síťových adres – například IPv4 adresy
- Typicky obsahuje menší množství portů s různými technologiemi
- Může se jednat o HW zařízení, ale i software – např Linux jako síťový router
- Základní funkcí routera je směrovat provoz dle cílové adresy, ale může být i kombinován například s firewallem nebo switchem – L3 switche
- Může kombinovat různé připojení k síti
  - Metalické připojení
  - Optické připojení
  - Bezdrátové připojení ( WiFi )
  - Vytáčené připojení ( telefonní linka )
    - Modem může být součástí routera

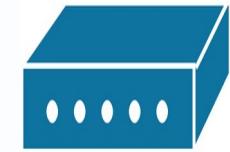
# Aktivní síťové prvky – Router -příklad



# Aktivní síťové prvky - Router -příklad zapojení



zdroj:<https://www.pearsonitcertification.com/articles/article.aspx?p=3129464&seqNum=3>

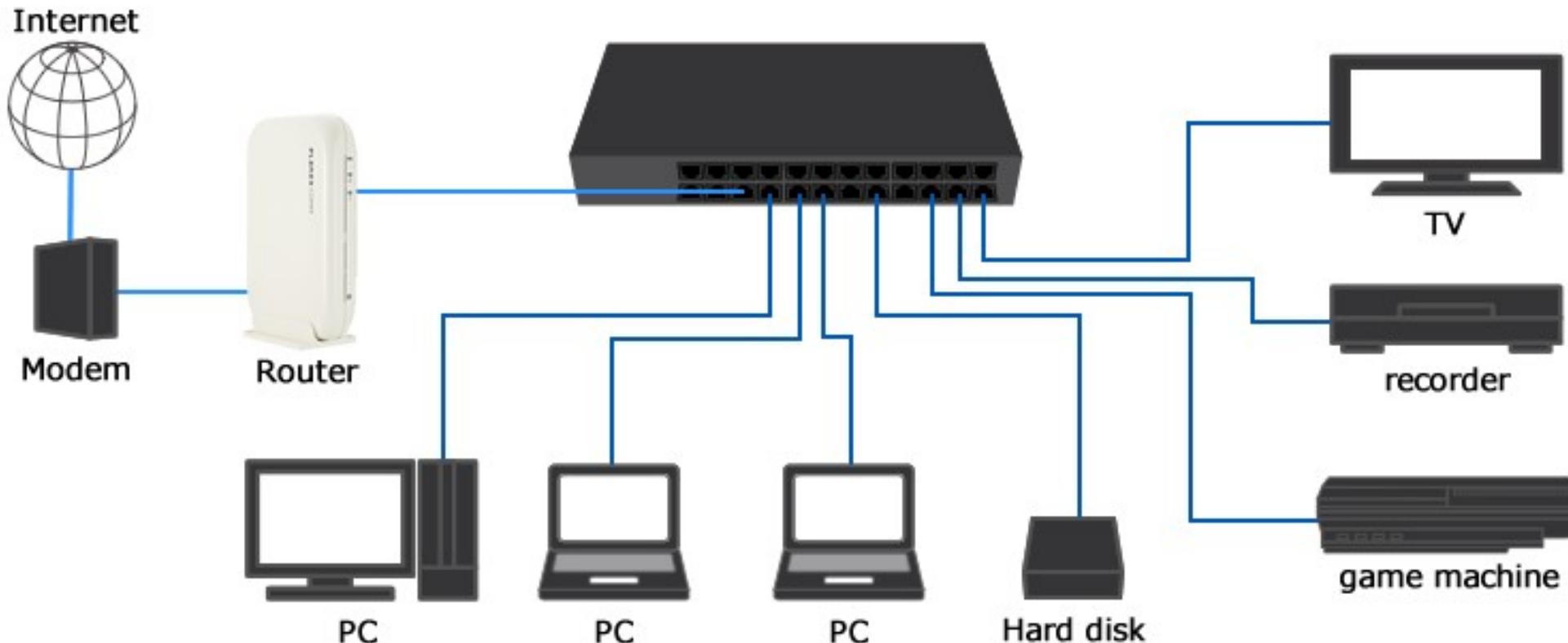
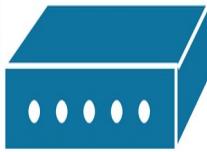


# Aktivní síťové prvky - Modem

- Modem - modulátor/demodulátor
- Zařízení umožňující realizovat přenos dat například telefonní linkou
- Může být jako samostatné zařízení nebo integrovaný v routeru či PC jako samostatná karta
- Dnes běžně používaný pro připojení pomocí ADSL, VDSL či pomocí kabelové televize
- Modem samotný pouze realizuje přenos dat( včetně autentizace a autorizace )



# Aktivní síťové prvky – Modem – příklad zapojení

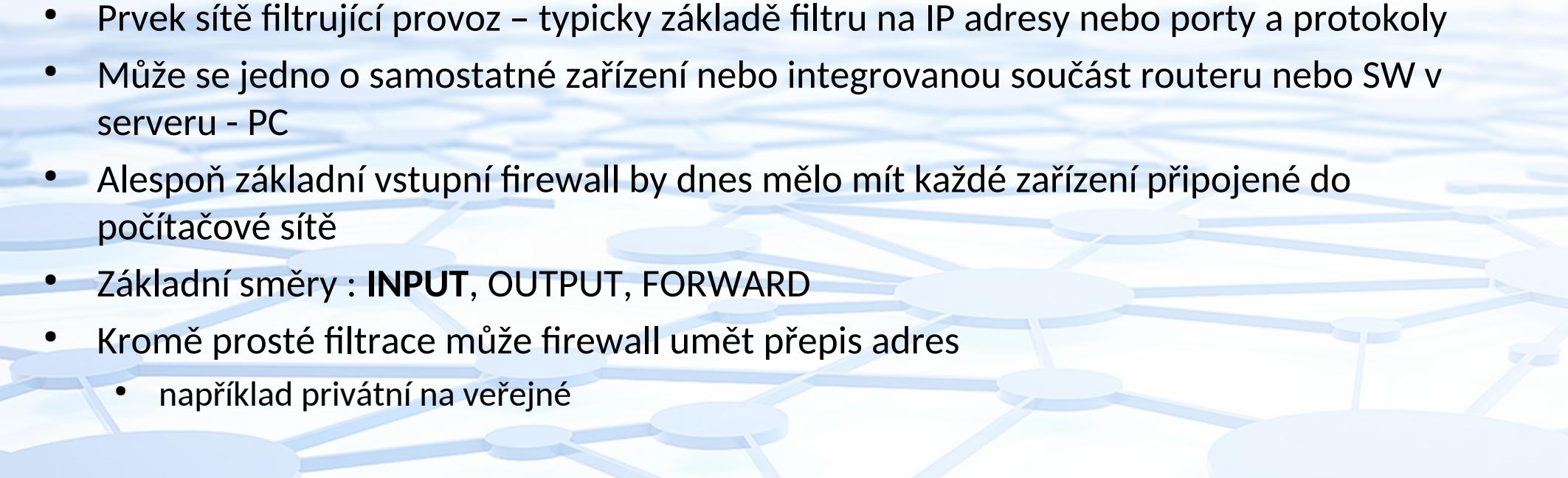


Zdroj: <https://www.fiber-optic-cable-sale.com/network-switch-router.html>

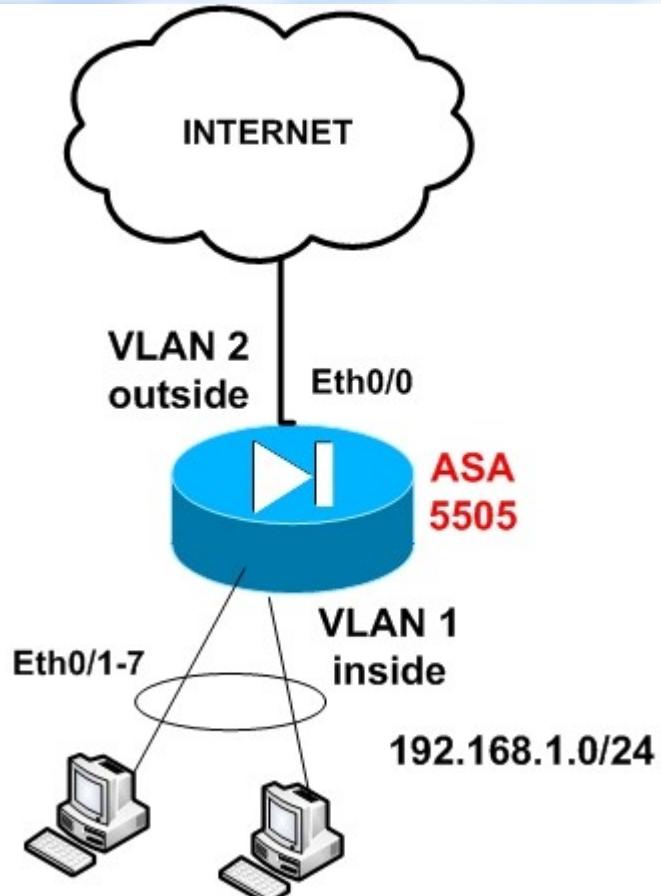


# Aktivní síťové prvky - Firewall

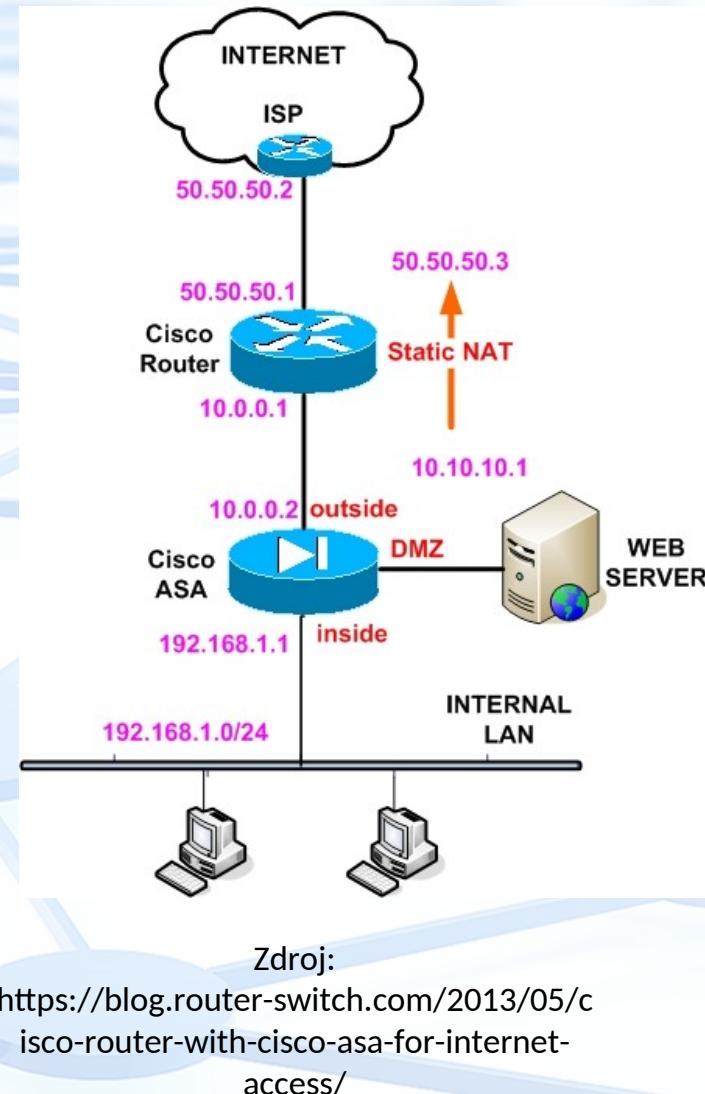
- Prvek sítě filtroující provoz – typicky základě filtru na IP adresy nebo porty a protokoly
- Může se jedno o samostatné zařízení nebo integrovanou součást routeru nebo SW v serveru - PC
- Alespoň základní vstupní firewall by dnes mělo mít každé zařízení připojené do počítačové sítě
- Základní směry : **INPUT, OUTPUT, FORWARD**
- Kromě prosté filtrace může firewall umět přepis adres
  - například privátní na veřejné



# Aktivní síťové prvky - Firewall - příklad zapojení



Zdroj:  
<https://blog.router-switch.com/2011/09/how-to-configure-cisco-asa-5505-firewall/>

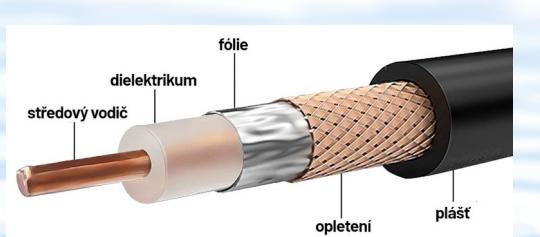


Zdroj:  
<https://blog.router-switch.com/2013/05/cisco-router-with-cisco-asa-for-internet-access/>

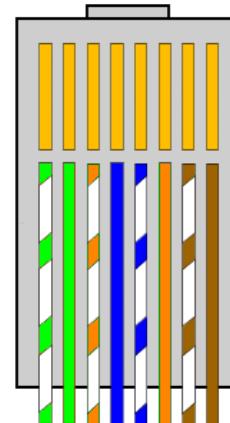
# Přenosová media - metalická



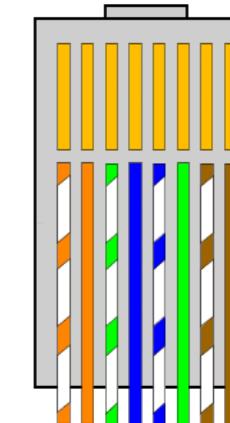
- Koaxiální kabel
  - Dva vodiče
    - Vnitřní stíněný drát
    - Vnější „válcový“ vodič - „opletení“
  - Dnes už v LAN sítích téměř nepoužívaný
    - Typickým použitím bylo pro Ethernet s maximální rychlosťí do 10Mbps
    - Síť byla sběrnicová a vyžadovala „terminátor“ - ukončení
  - Dnes používaný pro kabelové TV
    - A díky modemům se používá pro vysokorychlostní připojení prostřednictvím kabelové TV
- Nejběžněji používaný typ
  - Kroucené dvojlinky( Twist ):
    - RJ11 - 2 páry / 4 žíly, telefonní linky
    - RJ45 - 4 páry / 8 žil, počítačové sítě
      - Různé páry s různou délkou zkrutu
      - Dva druhy zapojení A a B ( změna křížení )
      - UTP - nestíněné vodiče ani celý kabel
      - STP - stíněný celý vodič a mechanicky odolnější
      - Označení kabelů CatX
        - Cat5, Cat5e, Cat6
        - Cat6a, Cat7,Cat8



T568A

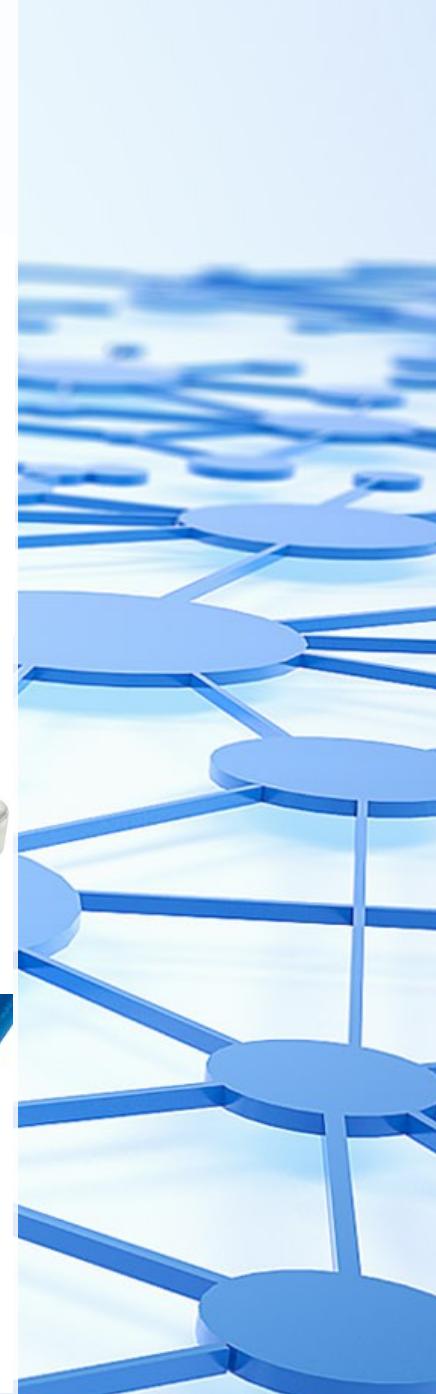


T568B



- 1Gbps

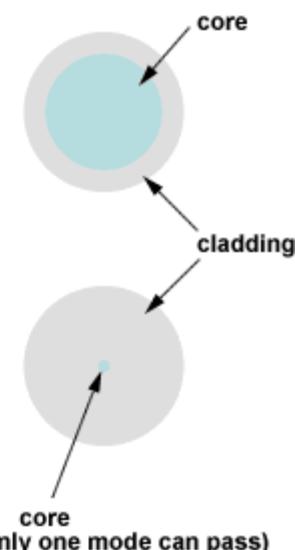
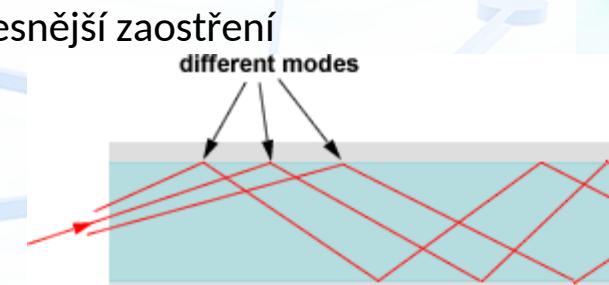
- 10Gbps



# Přenosová media - optická



- Přenášením signálem je světlo
- Vodič je tvořen optickým vláknem
- Dva druhy vláken
  - Jednovidové
    - přenos na delší vzdálenost – jednotky, desítky či stovky kilometrů
    - Dražší na pořízení, menší průměr, signál vede jen středem a umožňuje přesnější zaostření
    - Díky tomu dovoluje vyšší rychlosť na delší vzdálenost
    - Dovoluje přenos pouze jednoho signálu
  - Vícevidové / mnohavidové
    - přenos na krátké vzdálenosti – do cca 0,5 km
    - Levnější a odolnější než jednovidové
    - Silnější průměr a tím i větší časové rozostření
    - Nižší přenosová rychlosť a délka vedení
    - Dovoluje přenos více paralelních signálu ( vlnový multiplex )

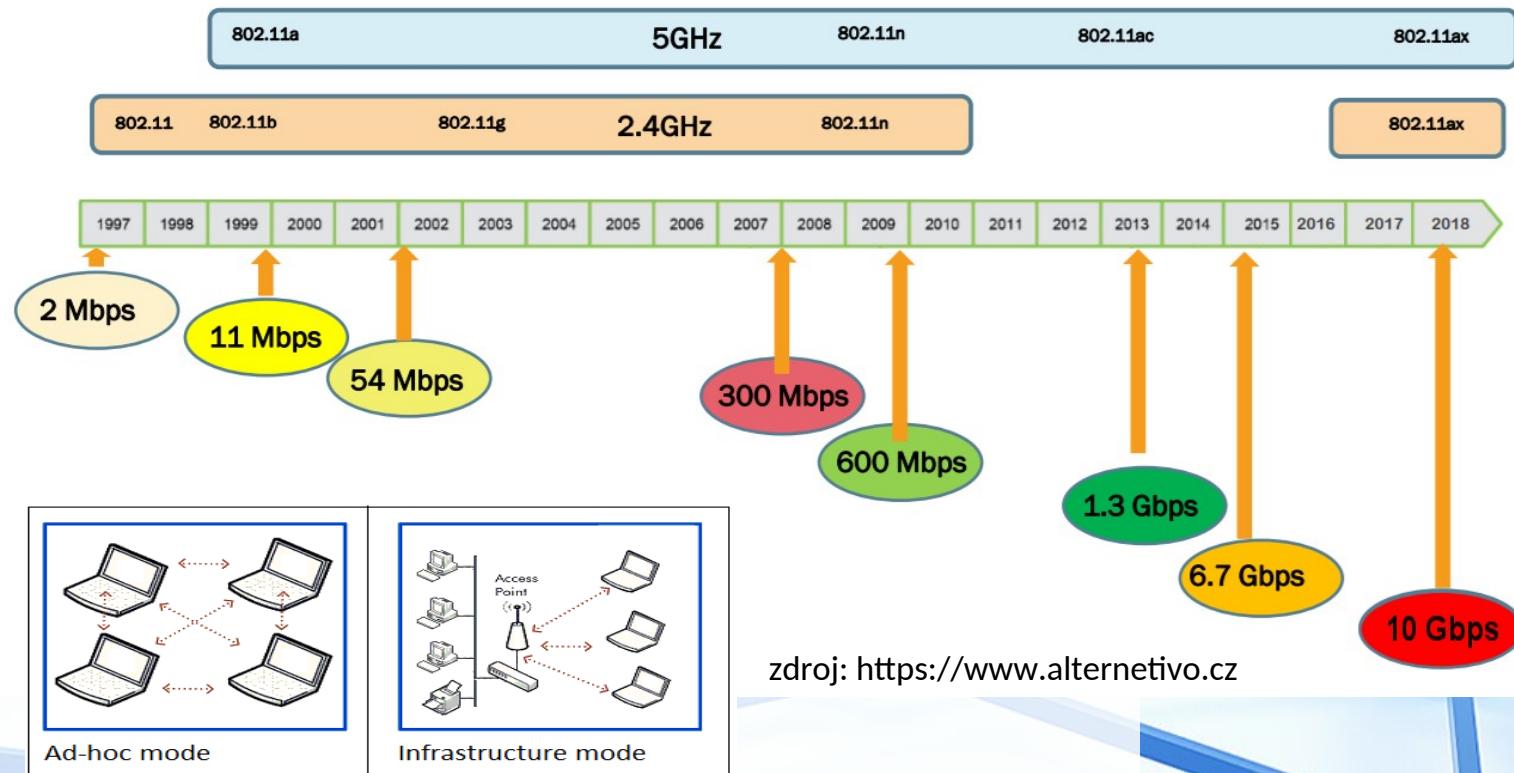


# Přenosová media: bezdrátové spoje



- Přenosové medium je volný prostor / vzduch
- Více typů bezdrátových přenosů:
  - Optický
    - Infráčervené spoje
    - Laserová pojítka
      - Typicky point-to-point
  - Radiové spoje - nejběžnější
    - Pro bezdrátové počítačové sítě se používá standard 802.11
    - Dva režimy
      - Ad-hoc – klienti se spojují mezi sebou
      - Infrastruktura – jeden společný přístupový bod

zdroj: <http://www.ecsystem.cz/vyrobky/uspesne-instalace/opticke-pojitko-pro-isp>



zdroj: <https://www.researchgate.net>

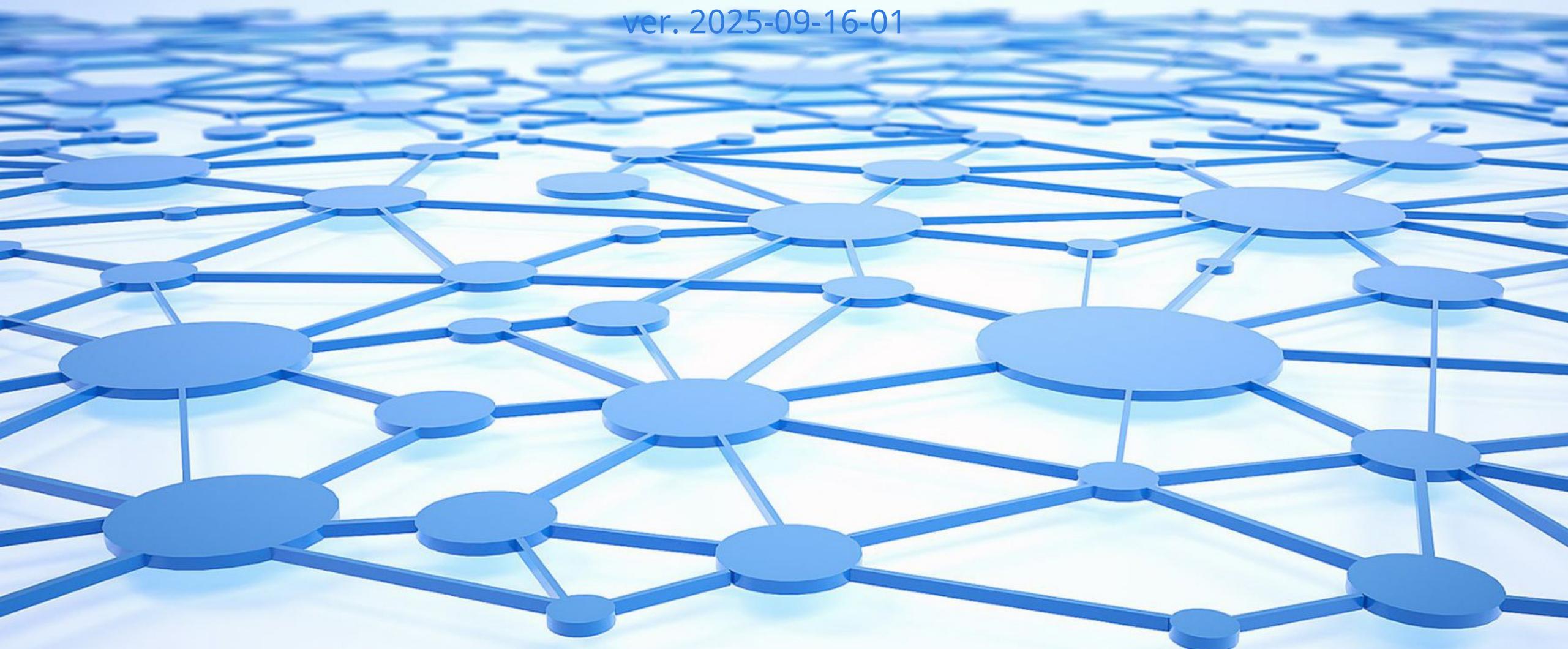
zdroj: <https://www.alternetivo.cz>

# Úvod do počítačových sítí

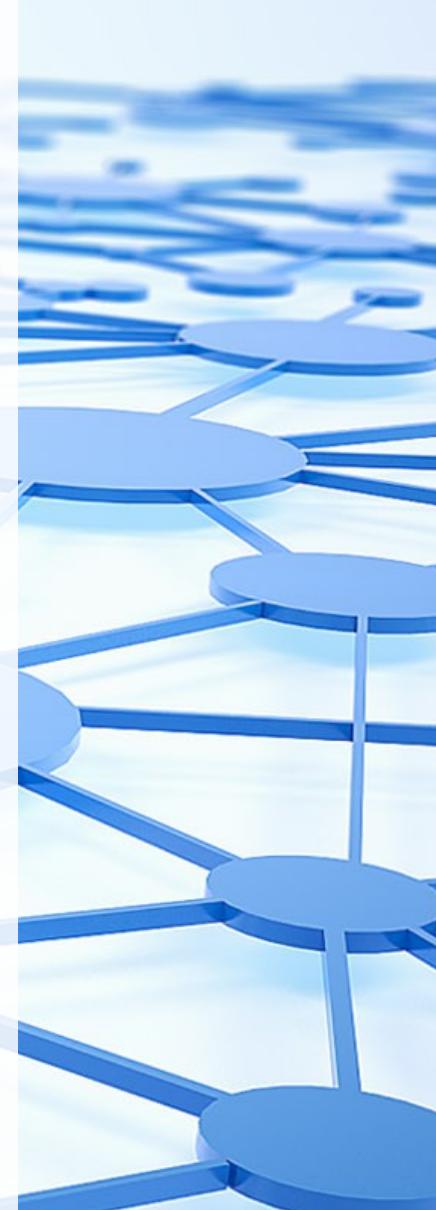
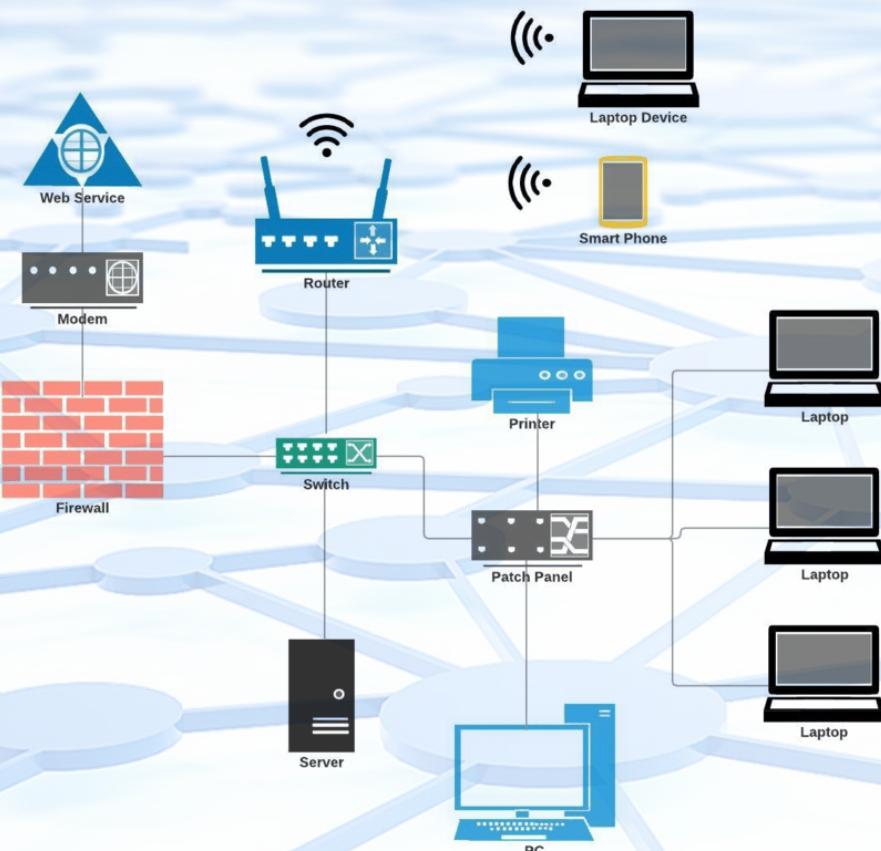
Přednáška 2

( 2025/2026 )

ver. 2025-09-16-01



# Opakování: Běžná počítačová síť

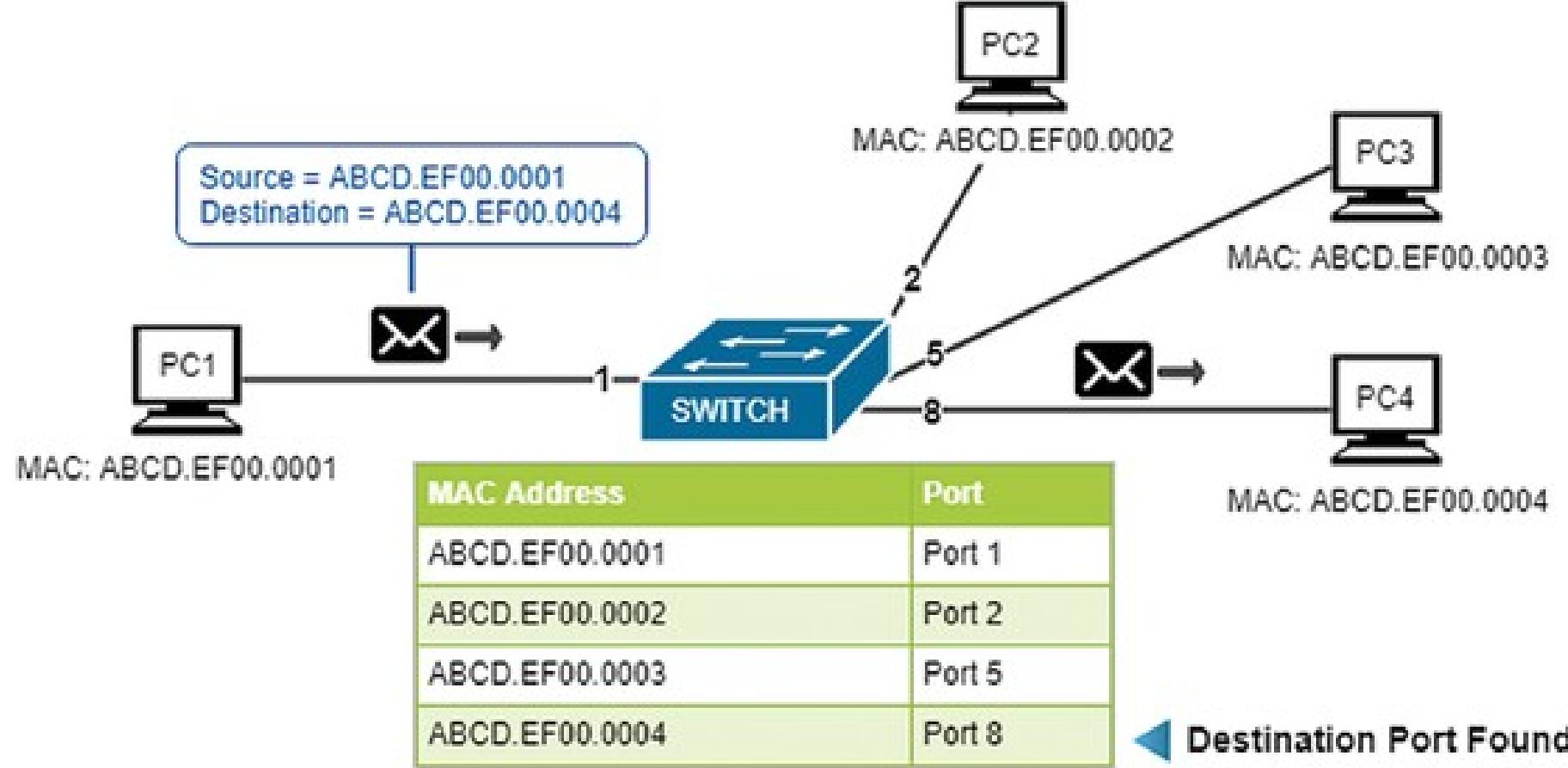


Zdroj: <https://www.itjones.com/blogs/2020/3/15/how-to-build-a-computer-network-for-your-small-business-part-1-the-basics>

# Adresace zařízení v LAN ( L2 )

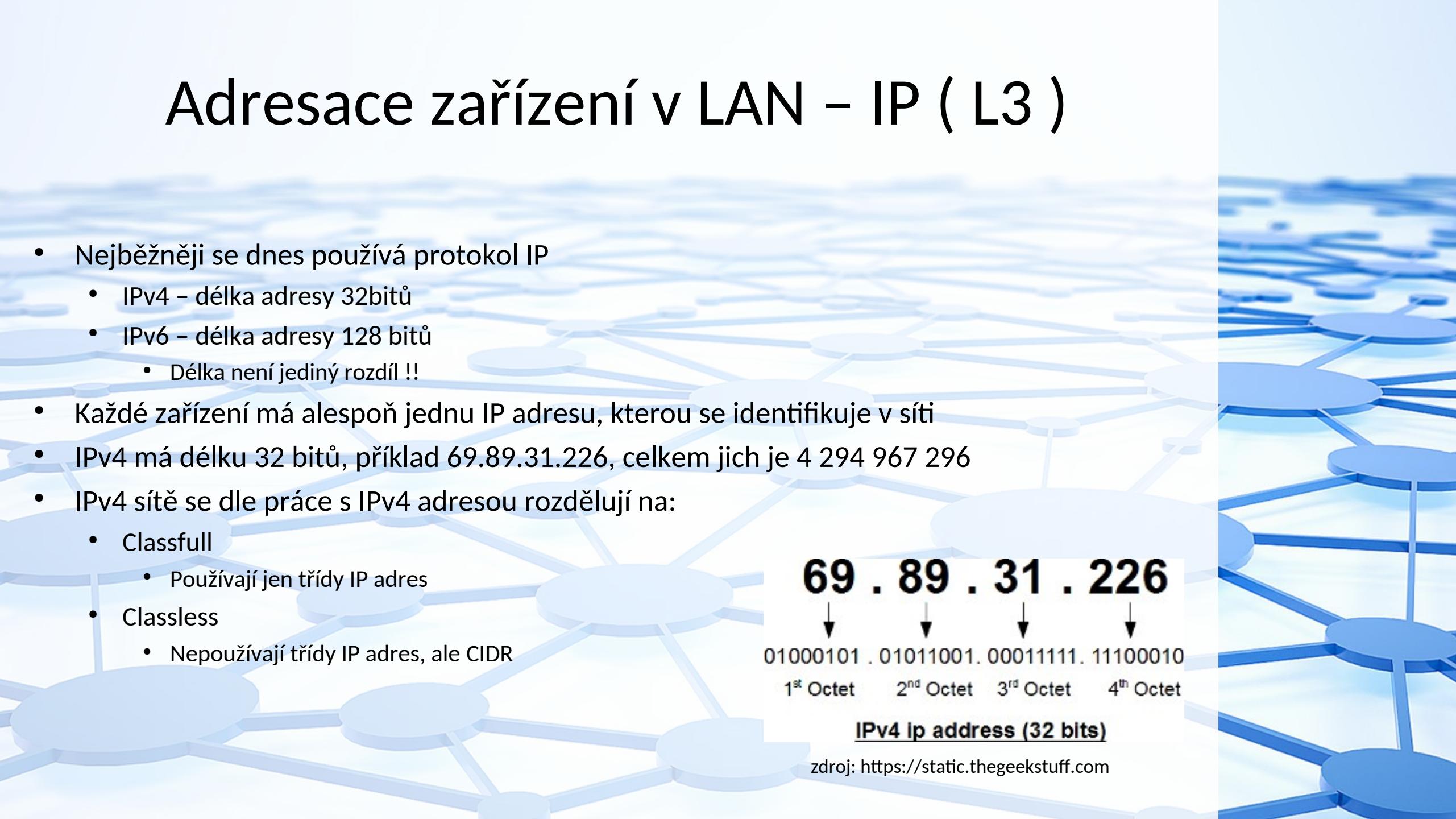
- Každé zařízení v LAN má svoji fyzickou adresu - MAC
- „Napevno“ spojena s konkrétním zařízením / síťovou kartou
  - Dnes už změnit jde, ale ve většině případů to není nutné ani vhodné
- V rámci jedné LAN musí být MAC unikátní
- Šest dvojciferných hexa čísel( 48 bitů ) – např 00:08:60:00:63:c9
  - První tři dvojice se označují jako VendorID a identifikují výrobce
    - **d0:94:66:19:e7:8d** → Dell Inc.
  - Další tři dvojice jsou volitelné výrobcem – ale stále v rámci LAN musí být unikátní
- Pokud není v rámci LAN dodržena unikátnost MAC adres provoz začne „flapovat“
  - Část zařízení v síti komunikuje s jedním zařízením se stejnou MAC a jiná část s druhou
  - Tento stav se průběžně mění jak zařízení komunikují
- Speciální adresa – FF:FF:FF:FF:FF:FF
  - Jedná se o L2 broadcast – rámeček s touto adresou bude doručen všem v LAN
  - V případě vytvoření smyčky může způsobit „broadcastovou“ bouři

# Adresace zařízení v LAN – tabulka adres pro switch



# Adresace zařízení v LAN – IP ( L3 )

- Nejběžněji se dnes používá protokol IP
  - IPv4 – délka adresy 32bitů
  - IPv6 – délka adresy 128 bitů
    - Délka není jediný rozdíl !!
- Každé zařízení má alespoň jednu IP adresu, kterou se identifikuje v síti
- IPv4 má délku 32 bitů, příklad 69.89.31.226, celkem jich je 4 294 967 296
- IPv4 sítě se dle práce s IPv4 adresou rozdělují na:
  - Classfull
    - Používají jen třídy IP adres
  - Classless
    - Nepoužívají třídy IP adres, ale CIDR



**69 . 89 . 31 . 226**

↓      ↓      ↓      ↓

01000101 . 01011001. 00011111. 11100010

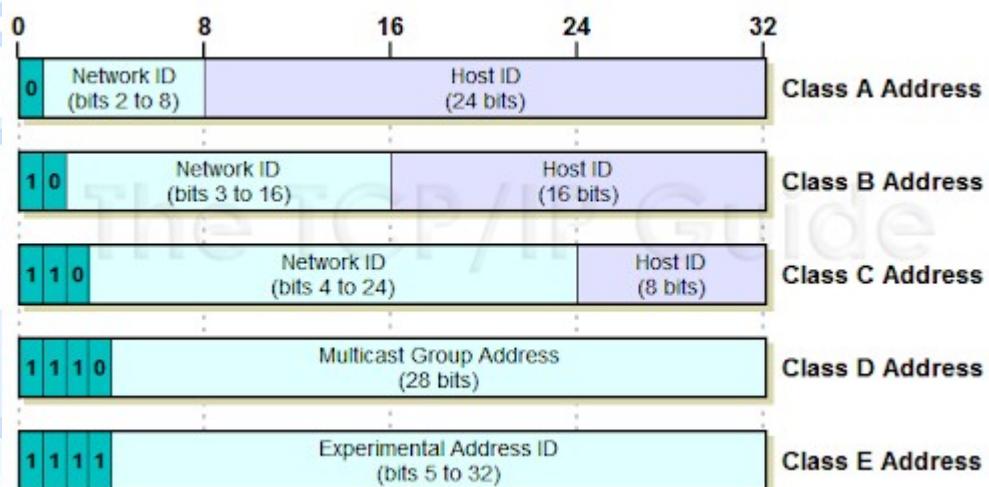
1<sup>st</sup> Octet    2<sup>nd</sup> Octet    3<sup>rd</sup> Octet    4<sup>th</sup> Octet

**IPv4 ip address (32 bits)**

zdroj: <https://static.thegeekstuff.com>

# IPv4 - třídy adres

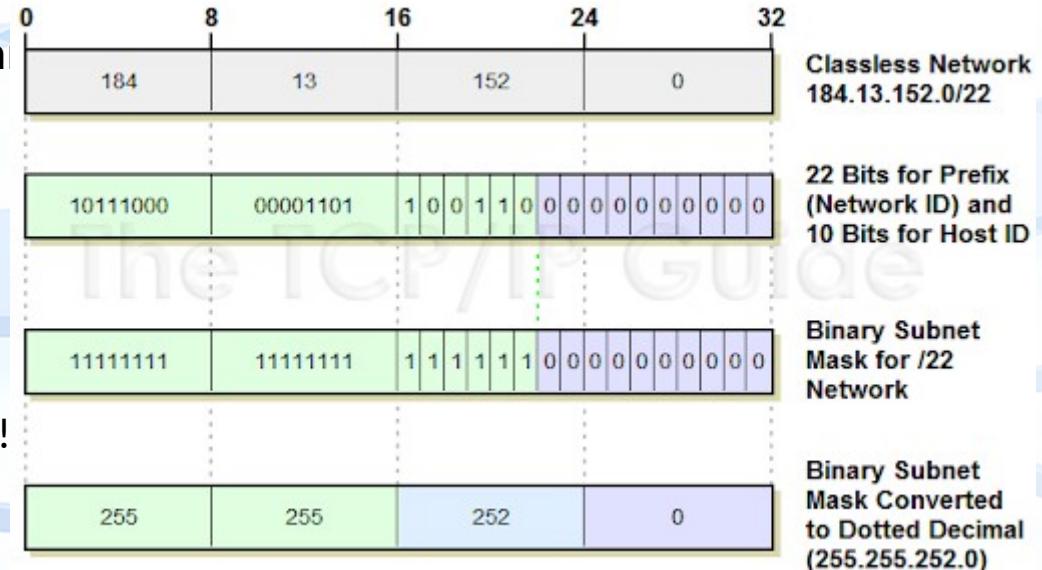
- Třídy adres definují adresní prostory a jsou přímo spjaté s „prefixem“ IP adresy
- Dnes už se příliš nepoužívá – pro běžný provoz, ale v terminologie ano
- Třída A (0.0.0.0 – 127.255.255.255)
  - Maska /8, až 16 777 214 stanic v každé síti
- Třída B (128.0.0.0 – 191. 255.255.255 )
  - Maska /16, až 65 534 stanic v každé síti
- Třída C (192.0.0.0 – 223.255.255.255 )
  - Maska /24, až 254 stanic v každé síti
- Třída D (224.0.0.0 – 239.255.255.255)
  - skupinové směrování - multicast
- Třída E (240.0.0.0 – 255.255.255.255)
  - Experimentální použití



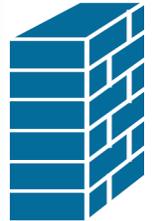
zdroj: <https://www.tcpipguide.com>

# IPv4 – CIDR

- CIDR - Classless Inter-Domain Routing, síť je daná maskou sítě, která není vázaná na IP adresu
- Dovoluje jemnější tvorbu podsítí
- Dnes již téměř kompletně nahradilo používání tříd
- Maska sítě - definuje kolik bitů zleva je pro danou síť fixních a kolik může libovolně použít
  - Spolu s IP a bránou tvoří základní konfigurační parametry IP sítí
  - Pomocí masky definuje:
    - Adresu sítě – nejnižší adresa v síti
      - $10.0.0.0/8 \rightarrow 10.0.0.0$
    - Adresu broadcastu – nejvyšší adresa v síti
      - $10.0.0.0/8 \rightarrow 10.255.255.255$
    - POZOR brána a maska spolu přímo nesouvisí !!
      - Na základě IP a masky nejde určit bránu



# IPv4 – privátní a speciální adresy



- Loopback
  - 127.0.0.1/8
  - Virtuální lokální interface
  - Je v každém zařízení a není nikdy dostupný z dalšího zařízení
- L3 Broadcast
  - 255.255.255.255
  - Stejně jako u L2 broadcastu ( FF:FF:FF:FF:FF:FF ), dochází k odeslání na všechna zařízení v LAN
- Privátní adresy
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - Adresy použité v lokálních sítích, které se následně pro provoz v Internetu překládají na veřejné pomocí NAT
  - Routery v internetu by měly blokovat provoz z veřejných IP na privátní
  - Tyto rozsahy nejsou nikde ve veřejném internetu použity
- IPv4 Prefix for Shared Address Space
  - 100.64.0.0/10
  - Adresy vyhrazené pro velké operátory a jejich interní komunikaci
  - Potřeba na interních sítích směrovat veřejné rozsahy

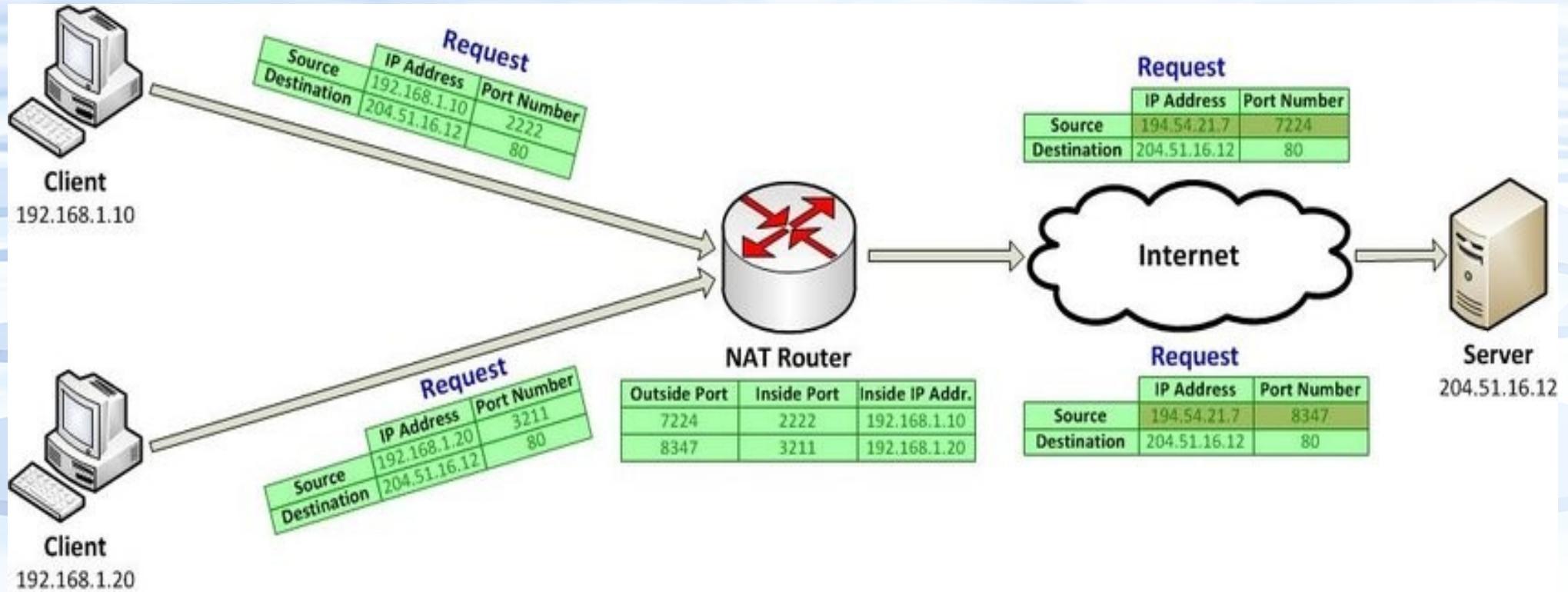


# IPv4 - NAT

- NAT - Network address translation
- Mechanismus umožňující překlad adres při přechodu firewallem/routerem
- Typicky se používá při překladu privátních adres na veřejné
  - Ale není to jediná možnost
  - Při přechodu NATem se zdrojová adresa nahradí veřejnou adresou routeru a zdrojový port náhodným VOLNÝM portem a toto mapování se zaznamená do NAT tabulky
- Pakety jdou internetem a zpět s adresou routeru, po návratu odpovědi na firewall/router je proveden zpětný překlad na původní hodnoty na základě údajů z NAT tabulky a paket je poslán dál do LAN



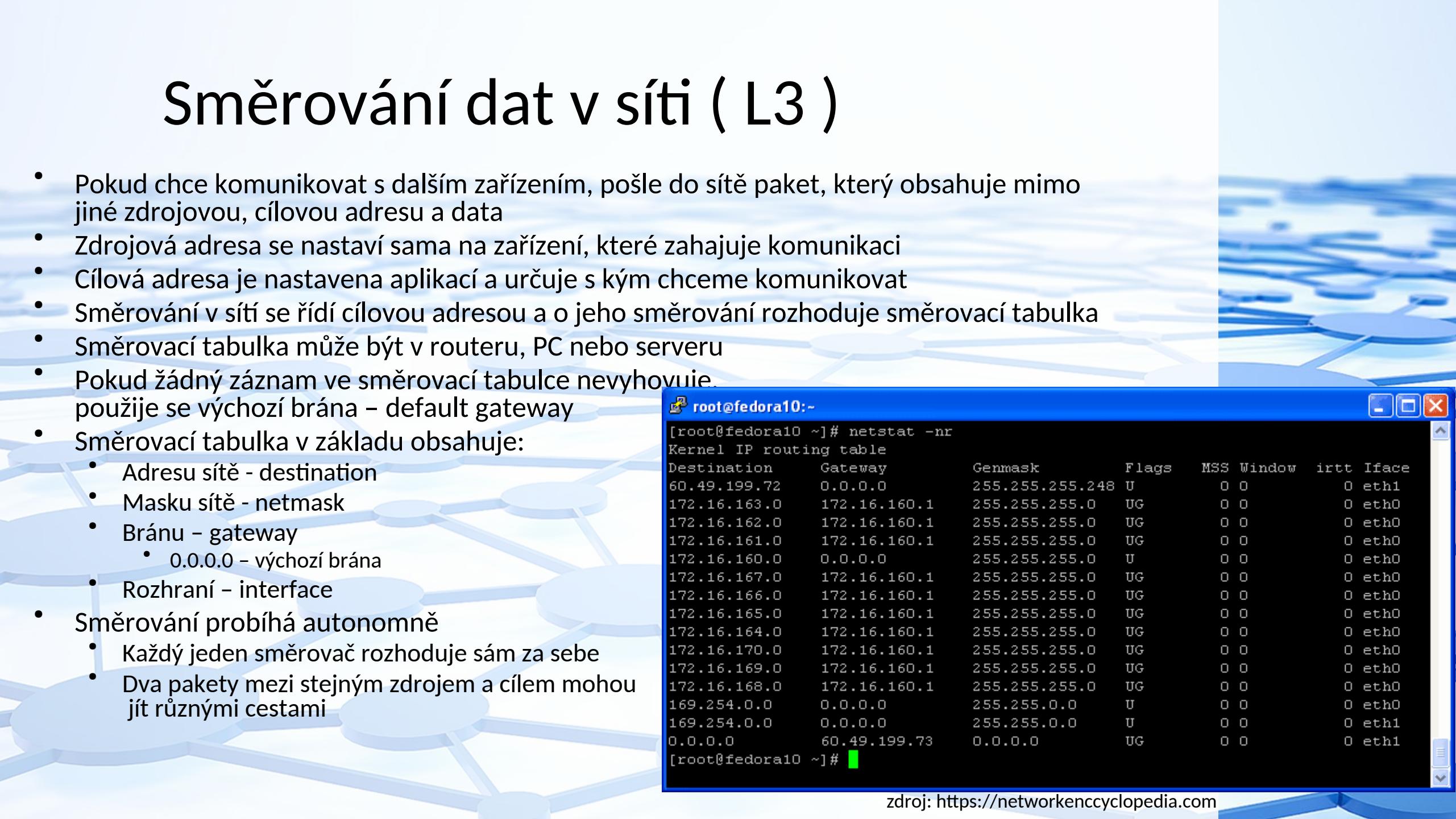
# IPv4 - NAT - příklad



Zdroj: [https://www.researchgate.net/figure/Network-Address-Translation-NAT-Working-Principle\\_fig4\\_334557400](https://www.researchgate.net/figure/Network-Address-Translation-NAT-Working-Principle_fig4_334557400)

# Směrování dat v síti ( L3 )

- Pokud chce komunikovat s dalším zařízením, pošle do sítě paket, který obsahuje mimo jiné zdrojovou, cílovou adresu a data
- Zdrojová adresa se nastaví sama na zařízení, které zahajuje komunikaci
- Cílová adresa je nastavena aplikací a určuje s kým chceme komunikovat
- Směrování v síti se řídí cílovou adresou a o jeho směrování rozhoduje směrovací tabulka
- Směrovací tabulka může být v routeru, PC nebo serveru
- Pokud žádný záznam ve směrovací tabulce nevyhovuje, použije se výchozí brána – default gateway
- Směrovací tabulka v základu obsahuje:
  - Adresu sítě - destination
  - Masku sítě - netmask
  - Bránu – gateway
    - 0.0.0.0 – výchozí brána
  - Rozhraní – interface
- Směrování probíhá autonomně
  - Každý jeden směrovač rozhoduje sám za sebe
  - Dva pakety mezi stejným zdrojem a cílem mohou jít různými cestami



```
root@fedora10:~# netstat -nr
Kernel IP routing table
Destination      Gateway        Genmask        Flags    MSS Window irtt Iface
60.49.199.72    0.0.0.0        255.255.255.248 U          0 0          0 eth1
172.16.163.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.162.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.161.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.160.0    0.0.0.0        255.255.255.0   U          0 0          0 eth0
172.16.167.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.166.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.165.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.164.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.170.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.169.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
172.16.168.0    172.16.160.1   255.255.255.0   UG         0 0          0 eth0
169.254.0.0     0.0.0.0        255.255.0.0     U          0 0          0 eth0
169.254.0.0     0.0.0.0        255.255.0.0     U          0 0          0 eth1
0.0.0.0          60.49.199.73   0.0.0.0       UG         0 0          0 eth1
[root@fedora10 ~]#
```

# „Základní/obslužné“ protokoly internetu

- Jedná se protokoly a na ně navázané služby, které často používáme „mimoděk“ a o jejich existenci často ani nevíme
- Mohou se na různých vrstvách sítě a využívat jak rámců ( např ARP, DHCP ), tak paketů ( DNS, ICMP )
- Příklady nejběžnějších jsou
  - ARP
    - Mapování MAC na IP a obráceně
  - DHCP
    - Automatická konfigurace sítě
  - DNS
    - Mapování doménových jmen na IP a obráceně
  - ICMP
    - Diagnostika a notifikace stavu sítě

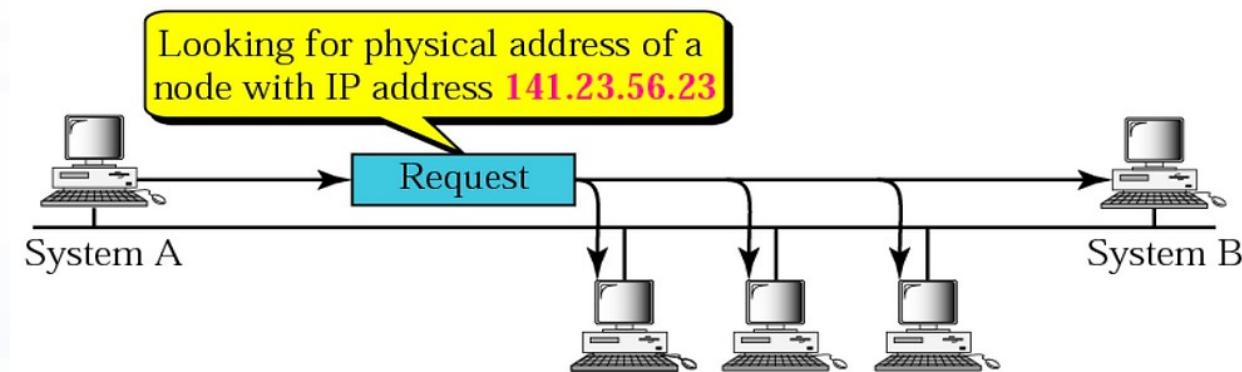
# Základní protokoly: ARP

- ARP - Address Resolution Protocol
- Protokol sloužící k zjišťování a mapování IP adresy na MAC a opačně
  - Nejčastěji je ARP vázán na IP a Ethernet, ale je postaven obecněji a může fungovat i v jiných sítích
  - Plní ARP tabulkou
- Záznam je do tabulky možné vložit ručně nebo ARP protokolem
  - Ruční záznam má typicky neomezenou platnost
  - Dynamický záznam je platný je krátký čas typicky 30-60s

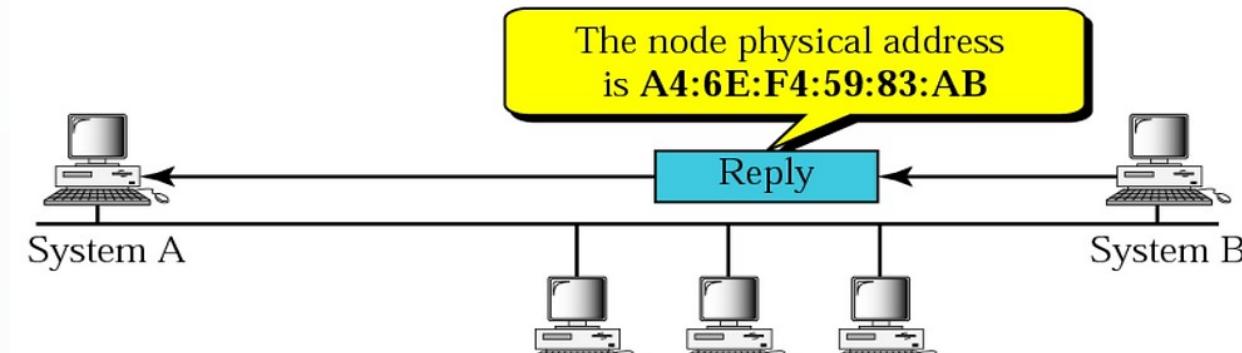
```
login as: admin
Using keyboard-interactive authentication.
Password:
Bad terminal type: "xterm". Will assume vt100.
Router>
Router>
Router>
Router> show arp-table
Address          HWtype  HWaddress           Flags Mask      Iface
210.210.3.1      ether    90:6C:AC:2D:DE:9A  C          wan1
10.10.10.40      ether    00:1E:67:52:E5:B6  C          lan1
10.10.10.50      ether    (incomplete)        C          lan1
10.10.10.30      ether    00:1E:67:52:E5:CE  C          lan1
10.10.10.45      ether    00:1E:67:52:E9:28  C          lan1
10.10.10.100     ether    00:1E:67:3B:1E:E0  C          lan1
10.10.10.120     ether    00:1E:67:3B:22:D2  C          lan1
210.210.3.39     *       <from_interface>   MP         wan1
Router>
```

# Základní protokoly: ARP - schéma

- Princip fungování
  - Pokud stanice potřebuje komunikovat s nějakou IP v LAN, pošle broadcastový rámec ( FF:FF:FF:FF:FF ) s dotazem na MAC pro IP, se kterou chce komunikovat
  - Všechny stanice v dané síti
    - limitované routerem
    - zprávu příjmou a stroj, který danou IP má odpoví tazateli dalším rámcem, kde je uvedena požadovaná MAC
- Může odpovědět i více stanic
  - to je problém a nastává pokud má více stanic stejnou IP
- Aby nebylo nutné dotazování před odesláním každého rámce ARP odpovědi se dočasně uchovávají v ARP tabulce
- Tato tabulka má dočasné záznamy a např po 30 jsou smazány a proces s opakuje



a. ARP request is broadcast

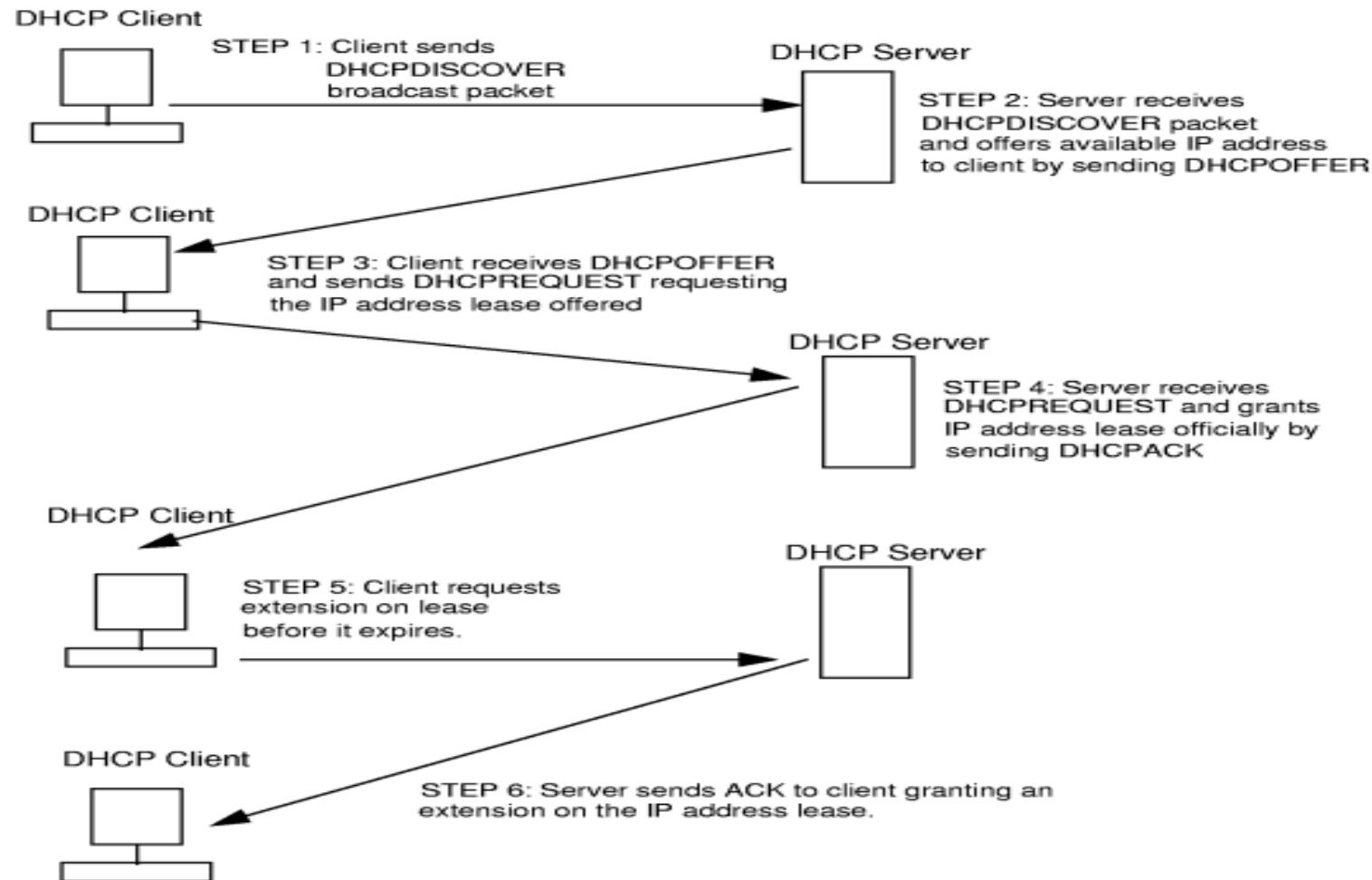


b. ARP reply is unicast

# Základní protokoly: DHCP

- DHCP - Dynamic Host Configuration Protocol
  - Vychází ze staršího BOOTP protokolu, se kterým není zpětně kompatibilní a který se dne už nepoužívá
- Možnost dynamického konfigurace síťových zařízení
- IP adresy je možné přidělovat ručně
  - V malých sítích typické a není s tím problém
- Problém nastává ve velkých sítích ( mnoho stanic ) a při častých změnách konfigurace sítě
  - administrativně náročné až nemožné
- Základní myšlenka je „auto“ konfigurace zařízení po připojení k počítačové síti
- Základní princip
  - Nemám IP, ale mám MAC => mohu komunikovat v rámci LAN ( broadcastem například )
  - Pošlu rámec s dotazem, zda někdo neví jak se mám na konfigurovat
  - Pokud v síti poslouchá nějaký DHCP server, dotaz přijme a v odpovědi pošle možnou konfiguraci
  - Konfigurace má omezenou platnost, po uplynutí poloviny intervalu se žádá o prodloužení
  - Ač to není běžné, odpověď může serverů více => vyberu si první odpověď
    - Typicky problém s defaultní konfigurací AP – dělají další „nepožadovaný“ DHCP server

# Základní protokoly: DHCP - princip komunikace



# Základní protokoly: DHCP – předávané informace

- Základní – potřebné pro fungování sítě
  - IP adresa
  - Maska
  - Brána
  - DNS server / servery
- Rozšiřující – nemusí být předávané, ale mohou zpřesňovat nastavení nebo konfigurovat další služby
  - NTP – časové servery
  - Wins, Active Directory doména – rozšířená konfigurace Windows stanic
  - NFS root / boot image – konfigurace pro boot bezdiskových stanic

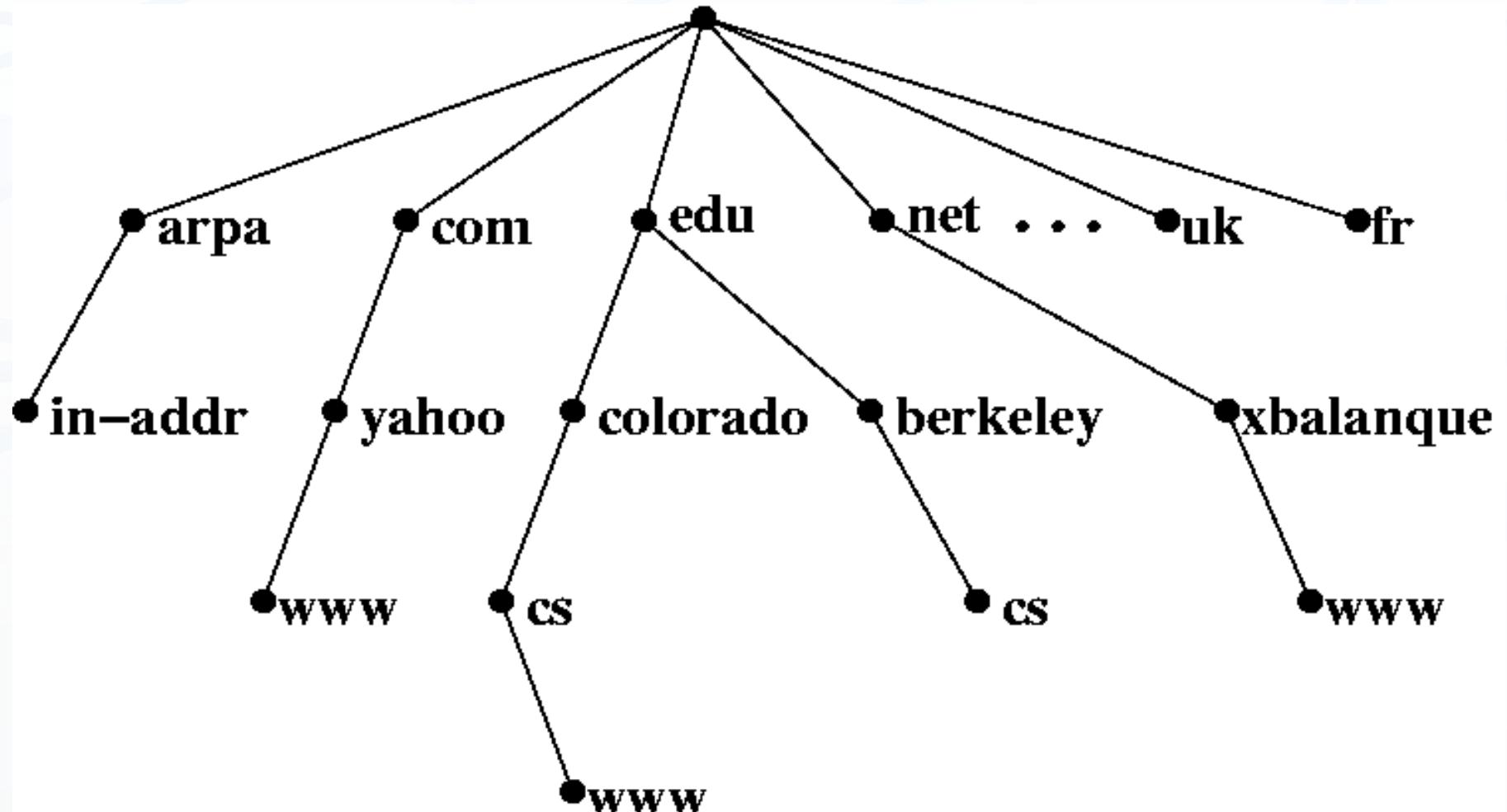
# Základní protokoly: DHCP – typy přidělené adresy

- Dynamické
  - Máme pool adres - 147.228.63.10 - 147.228.63.100 a z nich přidělíme jednu volnou
  - Do interní databáze se zaznamená MAC požadavku a přidělená IP
    - Pokud to jde, snažíme se stejné MAC přidělovat stále stejnou IP
    - Pokud volné IP dojdou jsou dočasné rezervace rušené a recyklované
      - Šetříme nutný počet adres – firma má 90 zaměstnanců a třísměnný provoz => potřebují naráz jen 30 IP
  - Typicky pro koncové stanice zařízení ,kde potřebuji IP, ale je jedno jakou
- Statická rezervace
  - Dynamické přidělování se nehodí pro zařízení poskytující obsah
    - Servery, tiskárny, ....
  - Do DHCP databáze provedu statickou rezervaci definicí vazby IP a MAC
    - Daná MAC VŽDY dostane stejnou IP
    - Staticky přidělené IP jsou vyjmuté z dynamického poolu

# Základní protokoly: DNS

- DNS - Domain Name System
- Slouží k převodu jména na IP a opačně
  - `www.kiv.zcu.cz => 147.228.63.11`
  - `147.228.63.11 => proteus.fav.zcu.cz` ( reverzní záznam )
- Jeden z „nejdůležitějších“ aplikačních protokolů Internetu
  - Pro většinu lidí platí, že pokud nejde DNS nejde Internet
- Ovlivňuje chování dalších protokolů a služeb ( např WWW nebo email )
- Některé služby se přes DNS i konfigurují ( DKIM, SPF, Windows Active Directory, ... )
- Jedná se o decentralizovaný systém
  - Výrazně se tím zvyšuje odolnost systému jako celku
- Hierarchický model oddělovaný „tečkami“ a začínající „neviditelnou“ tečkou vpravo
  - `www.kiv.zcu.cz.`
  - MAX 63 znaků na 1 úrovni, max. délka 255 znaků celkem, MAX 127 úrovní stromu

# Základní protokoly: DNS - schéma



# Základní protokoly: DNS – kořenové servery

- 13 kořenových jmenných( name ) serverů rozmístěných po celém světě
- Obsahují informace o kořenových doménách ( .cz .de .org .... )
- Kritické a velmi hlídané stoje
- Serverů není fyzicky 13, ale jsou spuštěny násobně v X instancích
  - Důvodem je redundancy, stabilita a rychlosť odezvy v dané lokalitě
- Kompletní seznam a info na <https://www.root-servers.org>
- Delegují odpovědi na jednotlivé správce národních a dalších domén
  - Například CZ.NIC pro .cz
- Sice se můžeme vždy ptát kořenových name-serverů, ale výhodnější je dotaz na nejbližší DNS – DNS ISP – který může odpovědět výrazně rychleji

# Základní protokoly: DNS - kořenové servery - mapa



zdroj: <https://coednssecurity.in>

# Základní protokoly: DNS – role serverů

- Primární
  - Obsahuje primární info o doméně
  - Slouží jako zdroj pro sekundární server
  - Je autoritativní pro své domény
- Sekundární
  - Přebírá info od doméně od primárního
    - Cyklicky nebo na vyzvání
  - Je autoritativní pro své domény
  - Neprovádí se zde změny záznamů
    - Aby nevznikal problém s konzistencí mezi primárním a sekundárním serverem
- Pomocný / cachovací
  - Nemá vlastní doménu
  - Slouží jako cache – snižuje datový trafik a zrychluje odpověď klientovi
  - Zná kořenové servery nebo referery, kterých se ptá pokud odpověď nezná
- Jedna instance může kombinovat všechny role / typy serverů

# Základní protokoly: DNS – typy záznamů

- DNS je vlastně databáze a má několik základních „datových“ typů
  - A
    - Obsahuje IP adresu, [proteus.fav.zcu.cz](http://proteus.fav.zcu.cz) => 147.228.63.11
    - Jedna IP by měla mít jen jeden A záznam
  - CNAME ( alias / přezdívka )
    - Odkazuje na jiný A nebo CNAME, [www.kiv.zcu.cz](http://www.kiv.zcu.cz) => [proteus.fav.zcu.cz](http://proteus.fav.zcu.cz).
  - MX ( mail exchange )
    - Slouží ke směrování pošty [zcu.cz](mailto:zcu.cz) => 10 fred.zcu.cz.
    - Nesmí ukazovat přímo na IP, obsahuje prioritu => záznamů může být více, nižší číslo má přednost
  - NS ( name server )
    - Autoritativní nameserver, [zcu.cz](http://zcu.cz) => [erebos.zcu.cz](http://erebos.zcu.cz).
  - PTR ( reverzní záznam )
    - Slouží k reverznímu mapování IP na jméno, 147.228.63.11 => [proteus.fav.zcu.cz](http://proteus.fav.zcu.cz).
  - AAAA
    - Stejně jako A, ale pro IPv6, [zcu.cz](http://zcu.cz) => 2001:718:1801:1058::1:100

# Základní protokoly: DNS – registrace domény

- Je třeba si zvolit registrátora – pro .CZ na <https://www.nic.cz/whois/registrar/>
  - Aktuálně jejich přes 40
  - Různé rozhraní, ceny, podmínky i nabízené domény
- Provede se ověření zda je doména volná v WHOIS databázi
- Potřebuje 2 „různé“ DNS servery
  - Dva kvůli redundanci – nemá smysl uvést jeden 2x, či dva ve stejné síti ...
  - Pokud vlastní DNS nemáte, většina registrátorů vám nabídne zdarma svůj
- Zvolíme platnost od 1 do 10 let – podle domény
- Vytvoří se žádost a přijde výzva k platbě
- Provedou se technické testy – ověření správného nastavení autoritatativních serverů
- Ověří se platba a doména se zveřejní
- Před expirací přijde typicky výzva k prodloužení – NATO POZOR, o doménu můžete při neuhranění přijít
  - Aktivní => Expirovaná => V ochranné lhůtě => Volná ....

# Základní protokoly: DNS – běžné komplikace

- Neaktuální záznamy na pomocných serverech
  - Typicky u velkých provoadrů s cílem šetřit přenosy
- Expirovaná doména v ochranné lhůtě
  - Doména je ještě vaše, ale už nefunguje
- Expirovaná doména po ochranné lhůtě
  - Obecně je doména volná a může si jí kdokoliv koupit
  - Může dojít ke spekulativnímu nákupu s cílem přeprodaje
- Podvržené nebo kompromitované DNS servery
  - Jeden z typu útoků v rámci Internetu
  - Oběť netuší, že nekomunikuje s pravým partnerem - například bankou
    - Může řešit DNSSec nebo SSL
    - Více na poslední přednášce o základech zabezpečení počítačových sítí

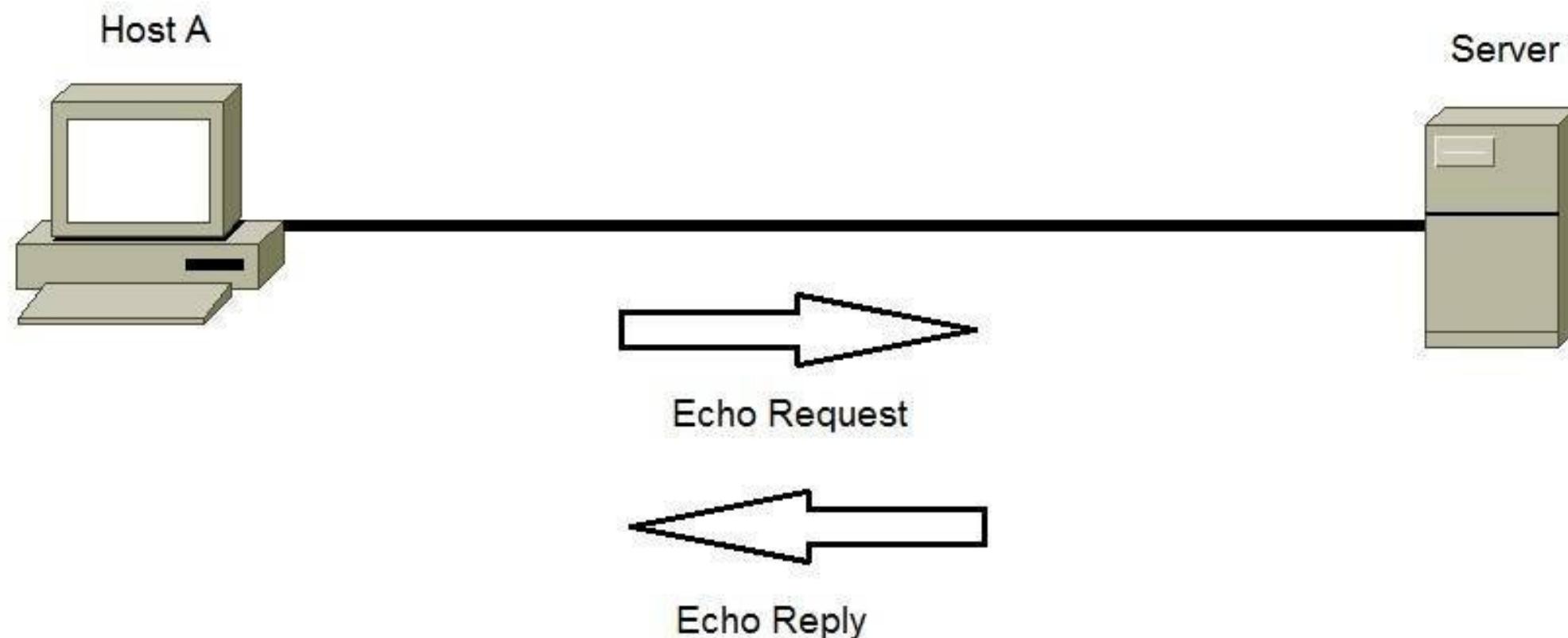
# Základní protokoly: ICMP

- ICMP Internet Control Message Protocol
- Pomocný / servisní protokol pro možnost informovat o chybových nebo nestandardních situacích
- Dnes použitelný ve dvou kontextech
  - Informace o chybách – např. Router zjistil, že nemůže data poslat na cílovou stanici, protože ta je nedostupná, tak pošle odesílateli zprávu o nedostupnosti cíle – aby se nečekalo
  - Diagnostika stavu sítě – cíleně posíláme zprávy s cílem zjistit dostupnost stroje nebo např uzly přes které v síti prochází
- Vyžívají jej programy jako ping nebo traceroute
- Na některých zařízeních se dle bezpečnostních pravidel zakazuje
  - Může zvýšit bezpečnost, protože útočník neví zda zařízení nežije či filtruje provoz
  - Podstatně hůře se diagnostikují závady
  - V případě problémů může systém mít výrazně delší odezvu, protože pokud nedostane info o chybě přes ICMP musí čekat na timeout

# Základní protokoly: ICMP – typy zpráv

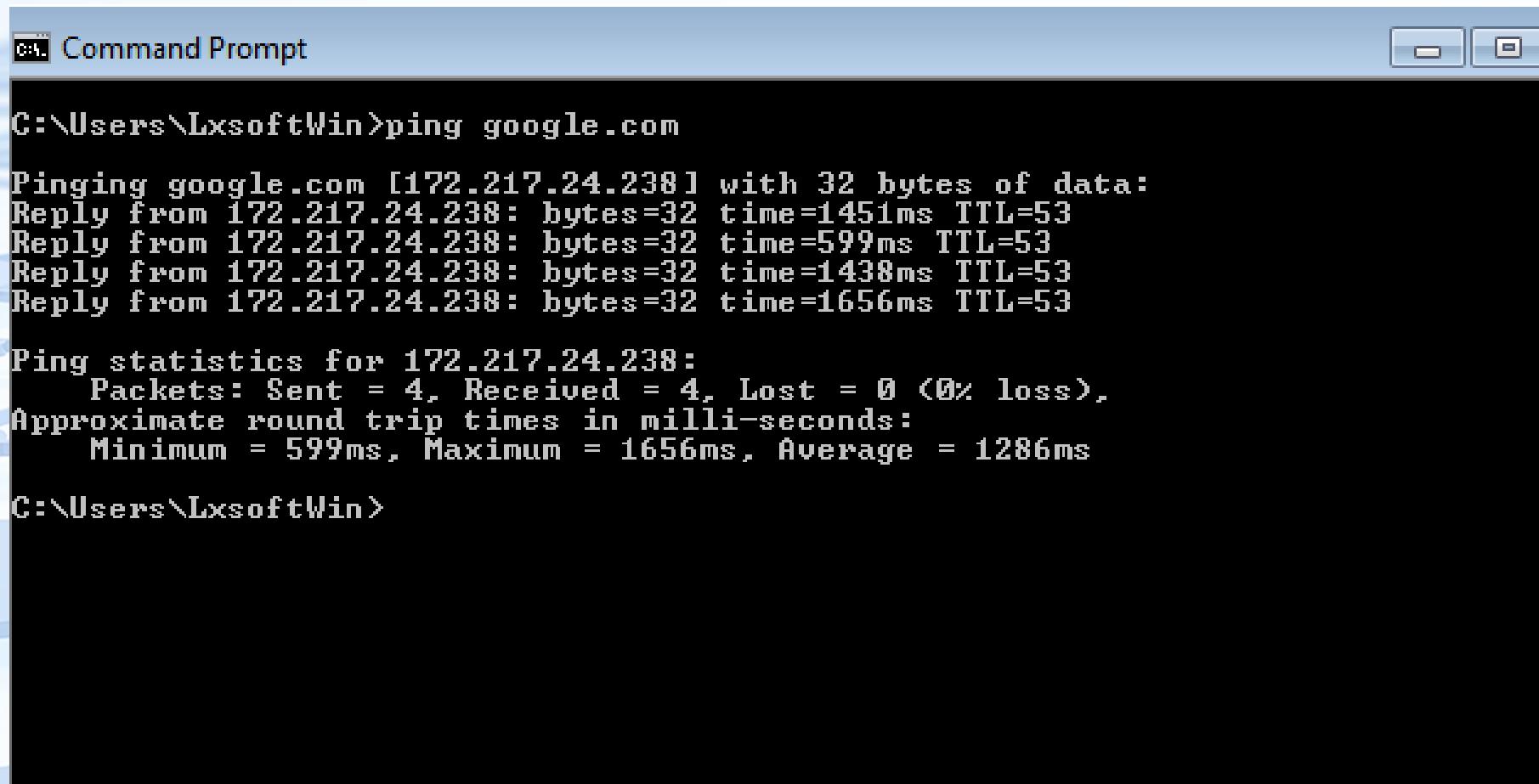
ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

# Základní protokoly: ICMP – ping



zdroj: <https://geek-university.com>

# Základní protokoly: ICMP - ping -výstup



```
Command Prompt

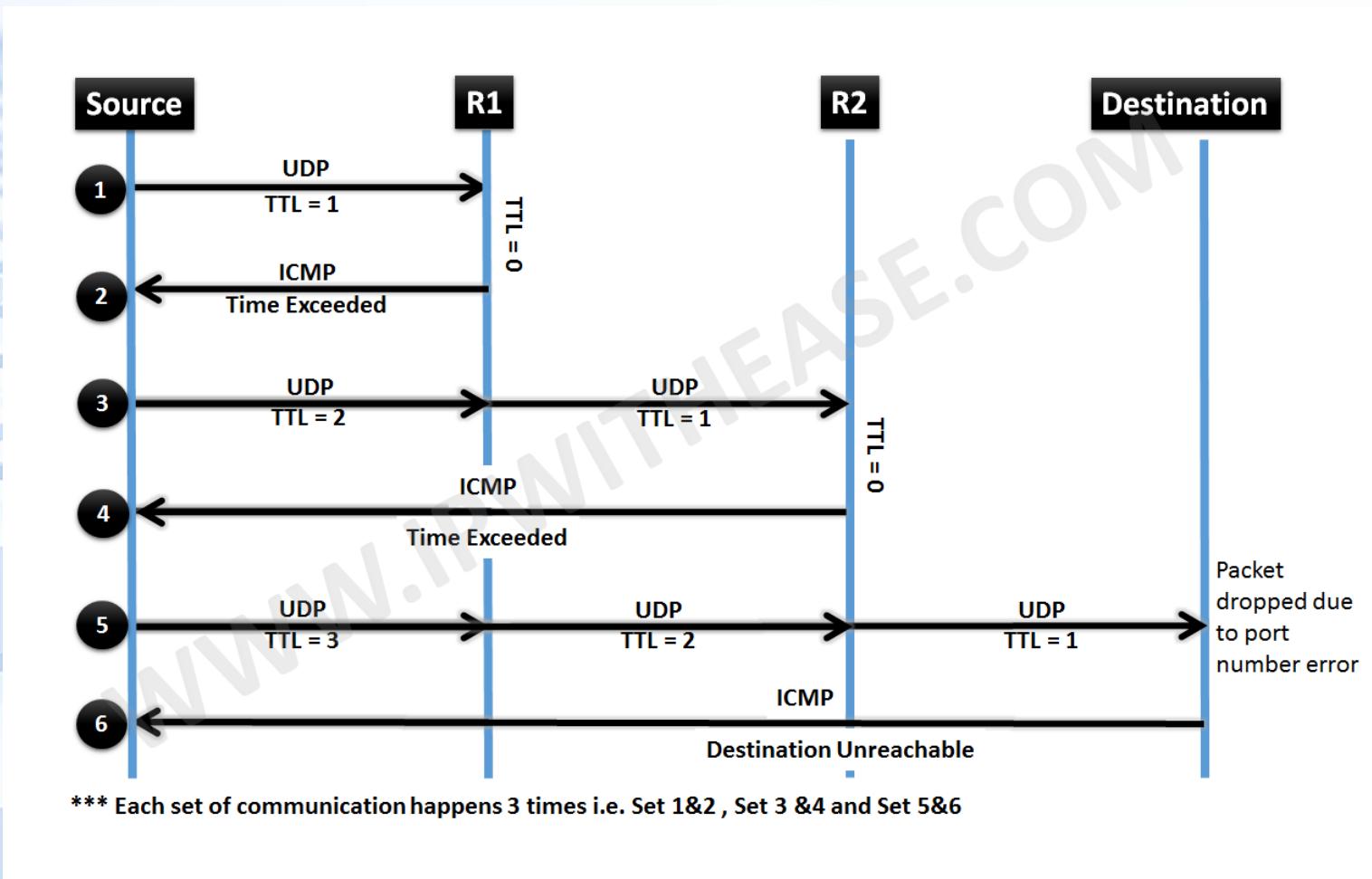
C:\Users\LxsoftWin>ping google.com

Pinging google.com [172.217.24.238] with 32 bytes of data:
Reply from 172.217.24.238: bytes=32 time=1451ms TTL=53
Reply from 172.217.24.238: bytes=32 time=599ms TTL=53
Reply from 172.217.24.238: bytes=32 time=1438ms TTL=53
Reply from 172.217.24.238: bytes=32 time=1656ms TTL=53

Ping statistics for 172.217.24.238:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 599ms, Maximum = 1656ms, Average = 1286ms

C:\Users\LxsoftWin>
```

# Základní protokoly: ICMP -traceroute



zdroj: <https://taylor.git-pages.mst.edu>

# Základní protokoly: ICMP -traceroute -výstup

```
C:\>tracert mediacollege.com
Tracing route to mediacollege.com [66.246.3.197]
over a maximum of 30 hops:
1 <10 ms <10 ms <10 ms 192.168.1.1
2 240 ms 421 ms 70 ms 219-88-164-1.jetstream.xtra.co.nz [219.88.164.1]
3 20 ms 30 ms 30 ms 210.55.205.123
4 * * *
Request timed out.
5 30 ms 30 ms 40 ms 202.50.245.197
6 30 ms 40 ms 40 ms g2-0-3.tkbr3.global-gateway.net.nz [202.37.245.140]
7 30 ms 30 ms 40 ms so-1-2-1-0.akbr3.global-gateway.net.nz [202.50.116.161]
8 160 ms 161 ms 160 ms p1-3.sjbr1.global-gateway.net.nz [202.50.116.178]
9 160 ms 171 ms 160 ms so-1-3-0-0.pabr3.global-gateway.net.nz [202.37.245.230]
10 160 ms 161 ms 170 ms pao1-br1-g2-1-101.gnaps.net [198.32.176.165]
11 180 ms 181 ms 180 ms lax1-br1-p2-1.gnaps.net [199.232.44.51]
12 170 ms 170 ms 171 ms lax1-br1-ge-0-1-0.gnaps.net [199.232.44.50]
13 240 ms 241 ms 240 ms nyc-m20-ge2-2-0.gnaps.net [199.232.44.21]
14 240 ms 251 ms 250 ms ash-m20-ge1-0-0.gnaps.net [199.232.131.36]
15 241 ms 240 ms 250 ms 0503.ge-0-0-0.gbri.ash.nac.net [207.99.39.157]
16 251 ms 260 ms 250 ms 0.so-2-2-0.gbr2.nwr.nac.net [209.123.11.29]
17 250 ms 260 ms 261 ms 0.so-0-3-0.gbri.oct.nac.net [209.123.11.233]
18 250 ms 260 ms 261 ms 209.123.182.243
19 250 ms 260 ms 261 ms sol.yourhost.co.nz [66.246.3.197]

Trace complete.
C:\>
```

zdroj: <https://www.tech-faq.com/>

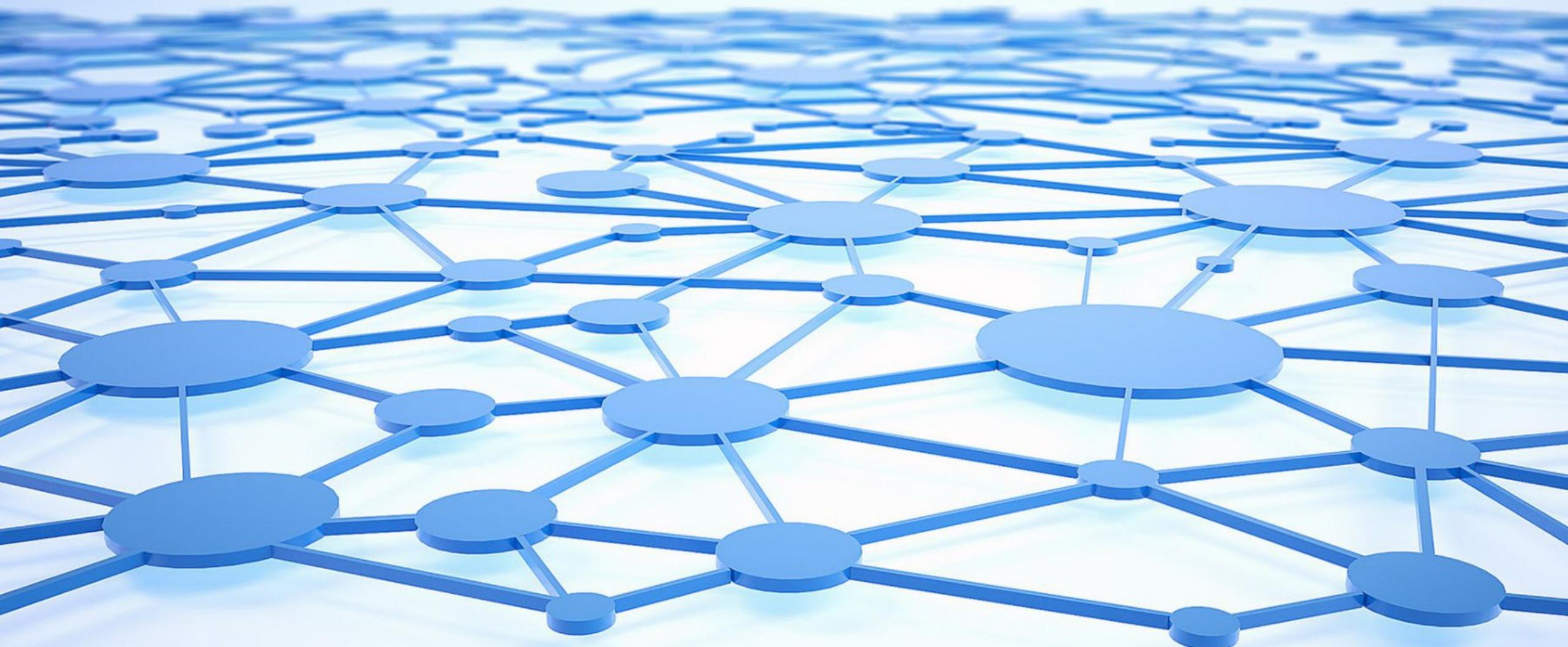
# Základní diagnostika sítě

- ipconfig / ifconfig / ip
  - Výpis nastavení síťového interfaceu
- route print / route / ip route
  - Výpis routovací tabulky
- arp
  - Výpis ARP tabulky
- ping
  - Ověření dostupnosti cíle pomocí ICMP
- tracert / traceroute
  - Zjištění trasy k cíli pomocí ICMP
- nslookup / dig
  - Zjištění hodnot z DNS
- netstat / ss
  - Výpis otevřených spojení
- wireshark / tcpdump
  - Sniffer síťové komunikace

# Úvod do počítačových sítí

Přednáška 3 ( 2025/2026 )

ver. 2025-09-23-01



# ISO/OSI model

- Referenční model popisující fungování počítačové sítě
  - Neřeší jednotlivé technologie, jen zavádí vrstvy a co by se na nich mělo řešit
- Má sedm vrstev
- Vznikl bez vazby na konkrétní implementaci
- Snaží se být maximalistický a řešit vše
  - Což se neukázalo jako ideální řešení
- Vychází spíše z telekomunikace
- Některé vrstvy jako L5 a L6 se nepoužívají
  - Respektive v praxi nejsou realizované jako samostatná vrstva
- Některé vrstvy se musí dále dělit L2
  - LLC A MAC

## ISO/OSI model

7	Data	Aplikační vrstva	Komunikace s procesem
6	Data	Prezentační vrstva	Prezentace dat a šifrování
5	Data	Relační vrstva	Koordinace komunikace
4	Segment	Transportní vrstva	Spojení
3	Paket	Síťová vrstva	Určení cesty a logická adresace
2	Rámec	Linková vrstva	MAC a LLC – fyzická adresace
1	Bity	Fyzická vrstva	Média, signál, binární přenos

# ISO/OSI model - popis vrstev

## 1. Fyzická

- Spojení dvou uzlů, přenos signálů jako je například světlo, řeší přenos jednotlivých bitů, řeší kódování, synchronizaci, časování, modulaci, parametry vodičů a signálu, konektory atd.

## 2. Linková

- dělí se vnitřně na dvě vrstvy
  - MAC - Media Access Control - fyzické adresování, přístup k mediu
  - LLC - Logical Link Control - zabezpečení proti chybám, multiplex, řízení toku dat

## 3. Síťová

- Přenos paketů - forwarding, adresace síťovou adresou, např. IPv4 nebo IPv6, směrování dat v síti - routing

## 4. Transportní

- „Přizpůsobovací vrstva“, přenos segmentů, adresace pomocí protokolu ( TCP, UDP, ICMP, .. ) a portu, spojuje aplikace

## 5. Relační

- Tvorba a ukončování spojení, tvoří relaci mezi komunikujícími uzly, řeší výpadek spojení, šifrování

## 6. Prezentační

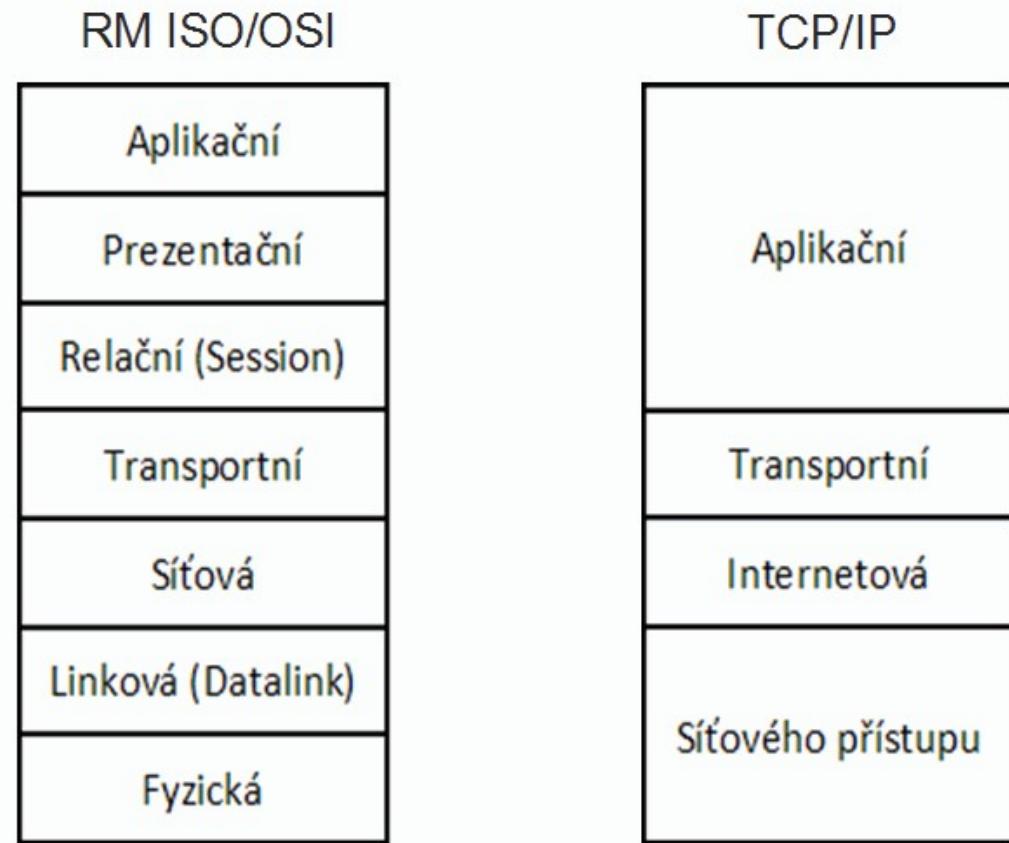
- Převádí přenesená data do formátu podporovaného aplikací a zpětně data od aplikaci do formátu vhodného pro přenos, měla by řešit rozdíly v platformách - například pořadí bitů v číslech, kódování atd.

## 7. Aplikační

- Vrstva určená pro běh samotných aplikací

# TCP/IP model

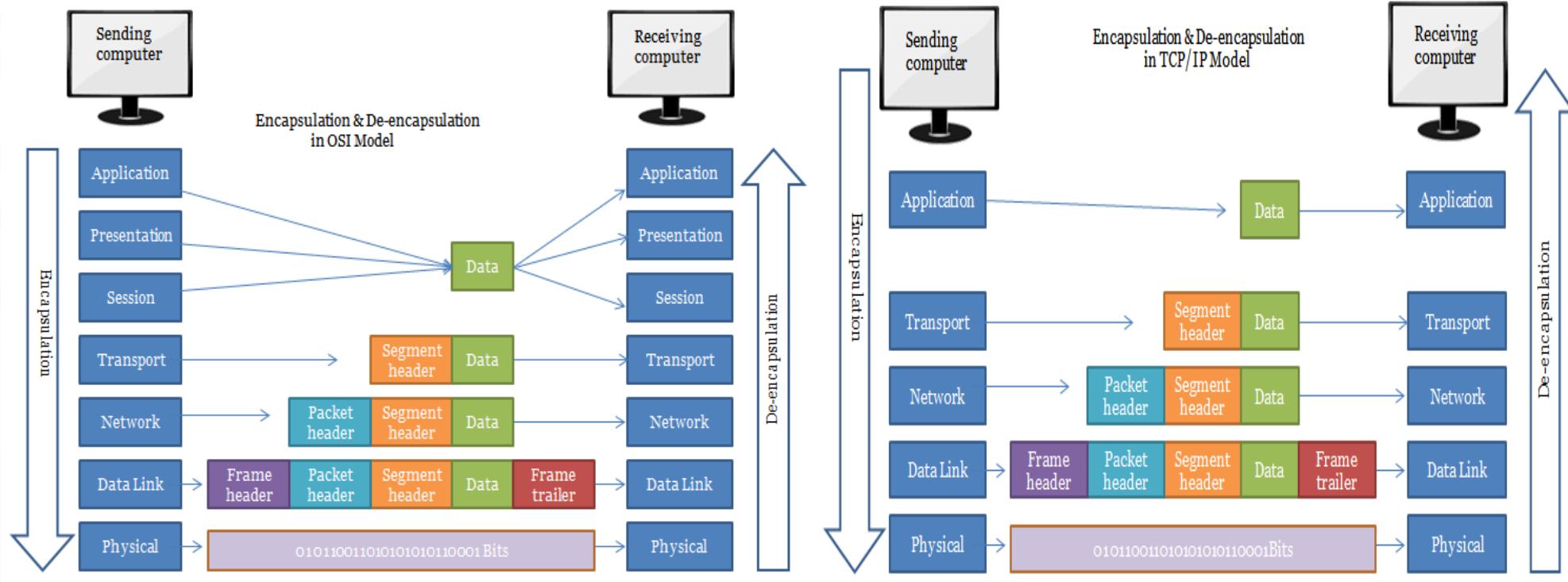
- ISO/OSI je příliš teoreticky, v praxi se používá zjednodušený TCP/IP model, který vznikl na základě použitých technologií
- TCP/IP model má jen 4 nebo 5 vrstvy:
  1. Vrstva síťového rozhraní – síťového přístupu
    - fyzická+přístupová (proto 4 nebo 5)
    - závislá na mediu, využívá existující řešení např Ethernet
  2. Síťová - internetová
    - nezávislá na mediu
    - řeší adresování a směrování, IP, ..
  3. Transportní
    - komunikace mezi procesy
    - adresa protokolem a portem, TCP, UDP, ..
  4. Aplikační
    - komunikace mezi aplikace
    - zde přenášíme ta dat co opravdu chceme SMTP, HTTP, FTP, ...



# Zapouzdření a rozbalování

- Nepřenášíme jen požadovaná data, ale i další řídící data, jako je například adresa odesílatele, adresa příjemce
- Na každé jednotlivé vrstvě ISO/OSI se adresuje jinak každá vrstva potřebuje přidat své řídící informace
- Zapouzdření
  - Přijmu data od nadřazené vrstvy ( například L2 přijme paket od L3 )
  - Přijatá data doplním další informace v podobě hlavičky a patičky
    - **L2 hlavička + „L3 data - paket“ + L2 patička**
  - Nově vzniklý datový blok předám nižší vrstvě ke zpracování
- Rozbalení
  - Přijmu data od nižší vrstvy ( například z L1 přijme blok dat, na L2 detekuje v nich rámec )
  - Odstraní z přijatých svou hlavičku a patičku
    - ~~L2 hlavička + „L3 data - paket“ + L2 patička~~
  - Předám data L3 vrstvě ke zpracování

# Zapouzdření a rozbalení - příklad



zdroj: <https://www.stackoverflow.com>

zdroj: <https://www.computernetworkingnotes.org>

# ISO/OSI – L1: Fyzická vrstva

- Základní funkce a význam fyzické vrstvy
- Zařízení fyzické vrstvy
- Topologie a typy spojů a typy přenosů
- Analogový a digitální signál
- Fourierova analýza
- Modulační rychlosť
- Nyquistovo a Shannonovo kriterium
- Rušení a útlum
- Synchronizace
- Kódování
- Přenos v základním a přeneseném pásmu
- Modulace
- Multiplex

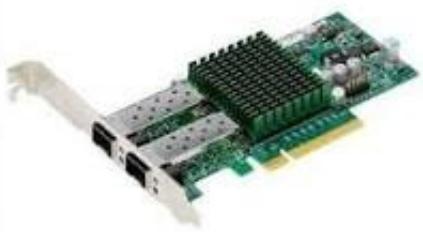
# L1: Základní funkce

- Jedná se nejnižší vrstvu, která je implementována hardwarově
- Hlavním cílem je přenos dat mezi dvěma koncovými body
- Chceme přenést data od nadřazené vrstvy ( frame/rámeček ) rozložený na jedničky a nuly
- K přenosu využíváme:
  - Zařízení: síťová karta, modem, opakovač, hub
  - Přenosové medium: metalický vodič, optické vlákno, vzduch/prostor, ...
  - Přenášený signál: fyzikálně měřitelný signál
    - napětí, proud, světlo, radiové vlny, zvuk, ....
- **Základní problém: Jak co nejpřesněji a nejrychleji přenést co největší množství binárních dat ( 0,1 ) pomocí výše uvedených signálů / vodičů / zařízení.**
  - Chceme přenášet co nejrychleji
  - Chceme přenášet co největší množství dat za jednotku času
    - V jednom stavu signálu nemusím přenést jen 0/1
  - Chceme přenášet s co nejmenším počtem chyb
    - Chyby „nepoznáme“ a musí je za nás řešit zabezpečení nebo vyšší vrstva, což „drahé“



# L1: Zařízení fyzické vrstvy

- Sítové karty ( vysílač – přijímač )
  - Metalické, optické, bezdrátové, ...
  - Zajišťují v zařízení ( jako je PC, switch, router, mobil, ... ) převod vstupního signálu na 0 a 1 a opačně
- Zesilovače
  - Přijímají signál a zesilují jej na výstupu
- Huby – Více-portové opakovače
  - Přijímají signál na jednom portu a kopírují/opakují jej na všechny ostatní porty
  - Zesilují signál a zároveň umožňují propojit více zařízení
- Modem – „modulátor - demodulátor“
  - Zajišťuje přenos dat počítačové sítě ( 0 a 1 ) jinou technologií – například prostřednictvím telefonní linky nebo kabelové televize
  - Musí umožnit jak přenos dat původního významu ( telefonní hovor ) tak nových – počítačové data
  - Opakem modemu je **kodek**, který zajišťuje přenos například hlasu pomocí počítačové sítě

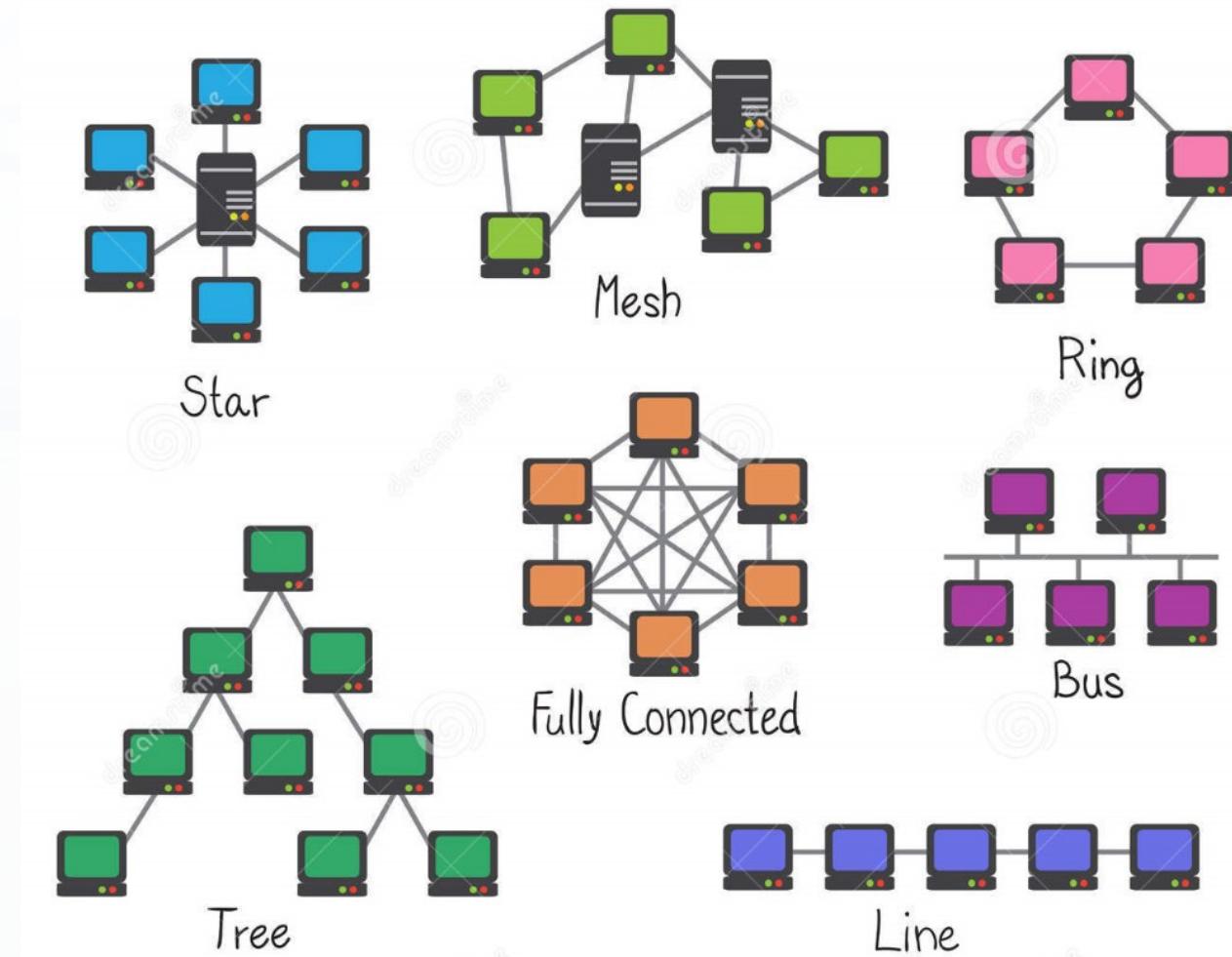


# L1: Typy propojení uzlů

- Propojení uzlů
  - Dvoubodové – jeden vysílač a jeden přijímač
    - Jednoduché, ale nákladné
    - Použití například v WAN
  - Mnohabodové – více vysílačů a více přijímačů
    - Nutné řešit sdílení media – tedy kdo kdy může vysílat aby nedocházelo ke kolizím
    - Dva typy z pohledu řízení přístupu
      - Pasivní – sběrnice, například Ethernet
        - Nikdo přístup k mediu aktivně neřídí, každé zařízení je zde samo za sebe
      - Aktivní – kruhové sítě, Token Ring
        - V síti existují zařízení či sdílené principy, které aktivně určují kdo kdy může síť využívat

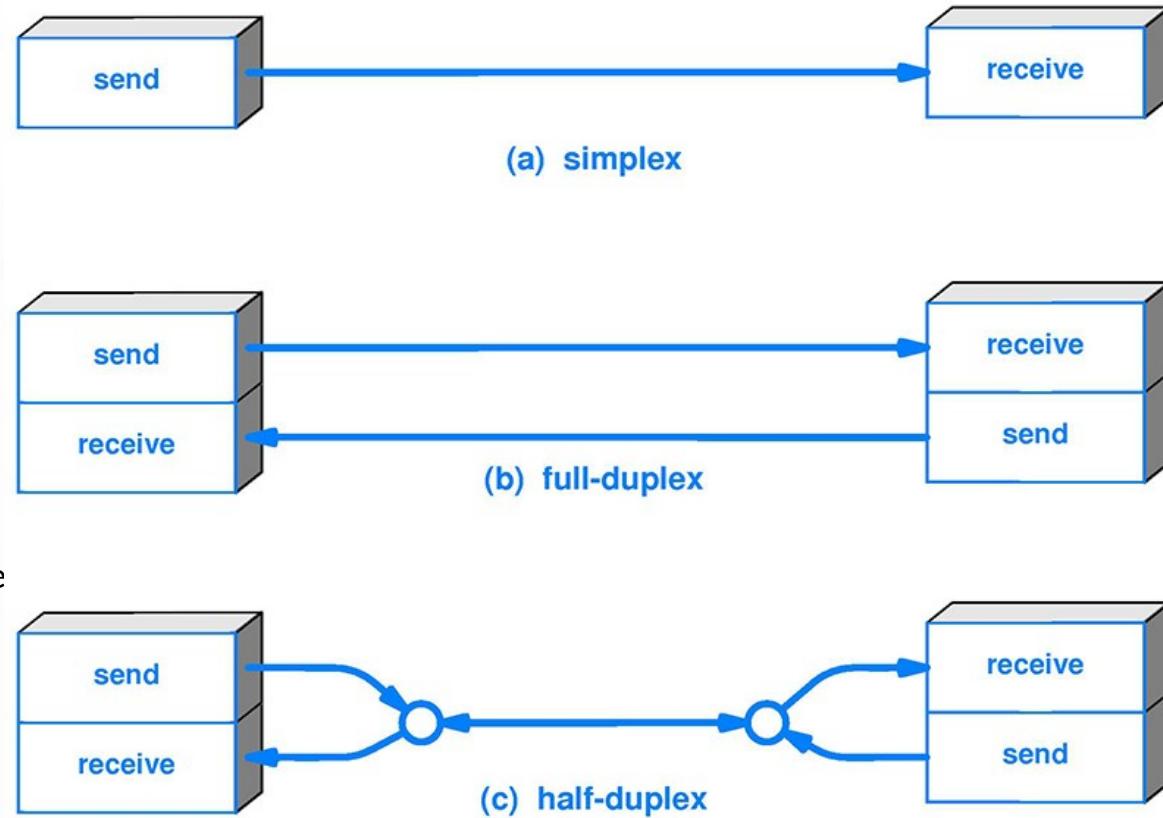
# L1: Topologie

- Topologie propojení
  - Lineární síť ( Line )
    - zapojení k sousedním uzelům vždy jedním spojem
  - Sběrnicové zapojení ( Bus )
    - všichni připojeni ke společné sběrnici
  - Kruhová síť ( ring )
    - „lineární síť“ s propojenými konci
    - Může být jak fyzický nebo logický například nad Bus
  - Hvězdová síť ( Star )
    - jeden hlavní uzel, ke kterému jsou připojeni ostatní
  - Hierarchická síť ( Tree )
    - stromové propojení
  - Neúplná polygonální síť ( Mesh )
    - propojení každého uzlu s jedním nebo více dalšími
  - Úplná polygonální síť ( Full Connected )
    - propojení každého uzlu s každým



# L1: Druhy linek dle možnosti přenosu

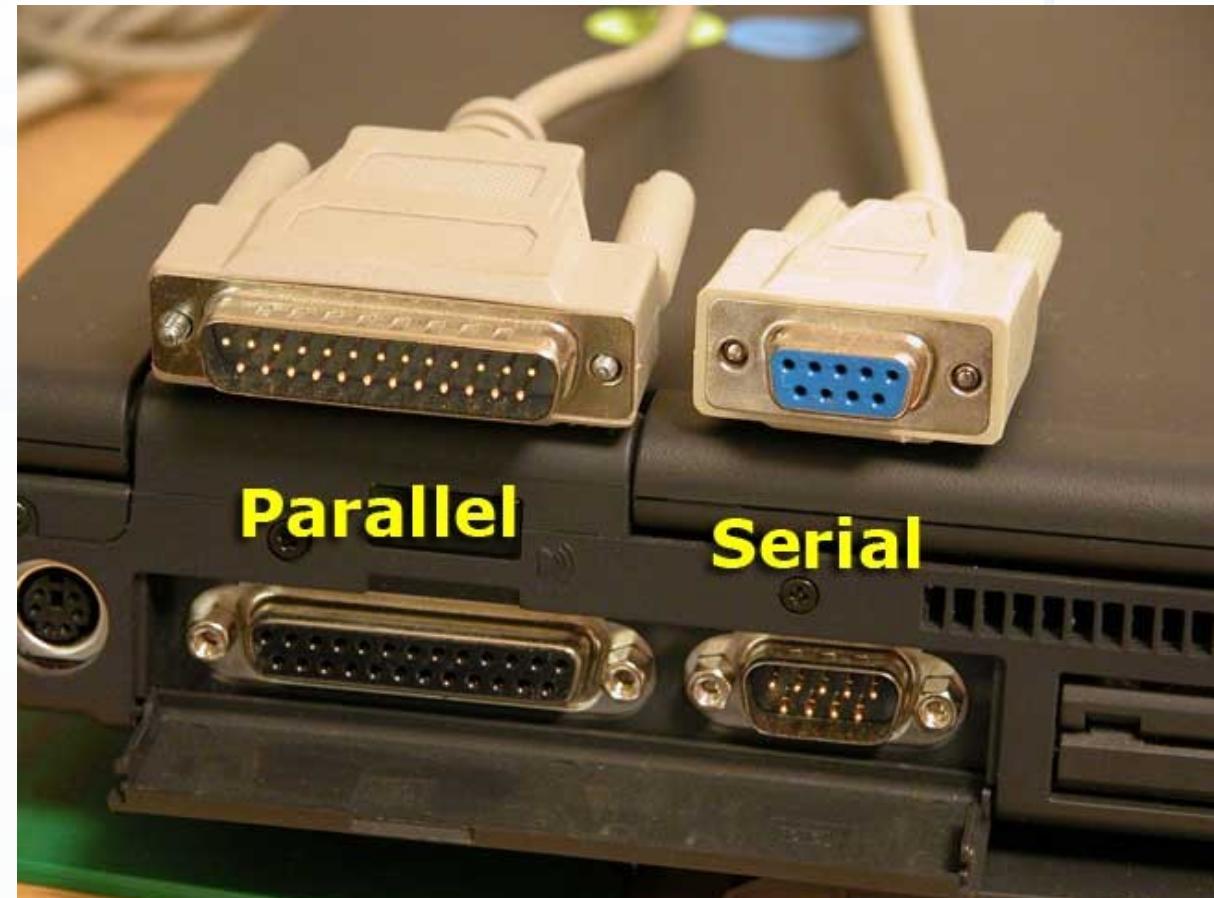
- Simplexní
  - přenos jen v jednom směru
  - Například TV
- Duplexní
  - současný přenos v obou směrech
  - Například Ethernet od 100Base-TX
    - extra vodič pro vysílání a příjem, které jsou navzájem křížené
- Poloduplexní
  - obousměrný přenos, ale NE současně
  - Například vysílačka, je nutné řízení
    - semafor – pomocí kterého se pozná kdo může vysílat, hrozí kolize



zdroj <https://www.blackbox.com>

# L1: Způsoby přenosu dat

- Paralelní
  - Přenášíme naráz více bitů prostřednictvím více vodičů
  - Typicky na kratší vzdálenosti
    - Například paralelní kabel k tiskárně
  - V počítačových sítích se příliš nepoužívá
- **Sériový přenos**
  - Přenášíme data bit po bitu jednou datovou cestou za sebou
  - Používám pouze jednu datovou cestu
    - Respektive mohou být dvě, pro vysílání a příjem
  - Typické pro počítačové sítě
  - Může dále dělit na
    - Synchronní
    - Asynchronní
    - Arytmický
  - **V dalších slidech budeme vždy uvažovat sériový přenos**



# L1: Analogový a digitální přenos

- Analogový přenos
  - Analogový signál, který je fyzikálně měřitelný či rozpoznatelný
    - Hodnota napětí, proudu, světla atd.
  - Je vždy specificky řešen pro dané prostředí a daný signál
  - **Může nabývat libovolných hodnot**
- Digitální přenos
  - Neřešíme konkrétní hodnoty veličin, ale stavů
    - **Může nabývat jen konkrétních stavů**
  - Tedy z vybraných hodnot analogového signálu odvodíme stav digitálního signálu
  - Přenášíme 0 a 1, ale můžeme je přenášet i více bitů naráz pomocí více stavů
    - Více stavů umožňuje přenést ne jen 0 a 1, ale např 00, 11, 01, 10, ...
  - Přenášenou jednotkou je 1 bit
  - Rychlosť přenosu je b/s - počet bitů za vteřinu

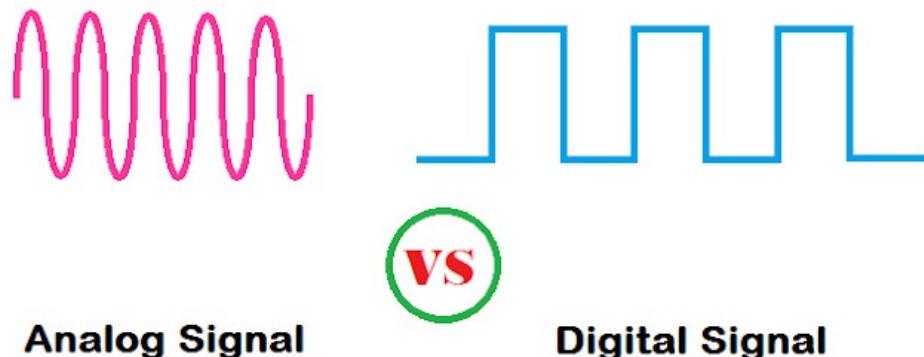
# L1: Reálný přenos

- Co přenášíme
  - Analogový signál
    - Harmonický
    - Libovolný
      - Problém s popisem
  - Rádi bychom obdélníkový signál
    - Defakto jedničky a nuly – dobře by se četl
  - Fourierův rozklad / transformace / analýza

$$f(x) = a_0 + \sum_{n=1}^{\infty} (a_n \cos \omega_0 t + b_n \sin \omega_0 t)$$

$$a_0 = \frac{1}{T} \int_0^T f(t) dt$$

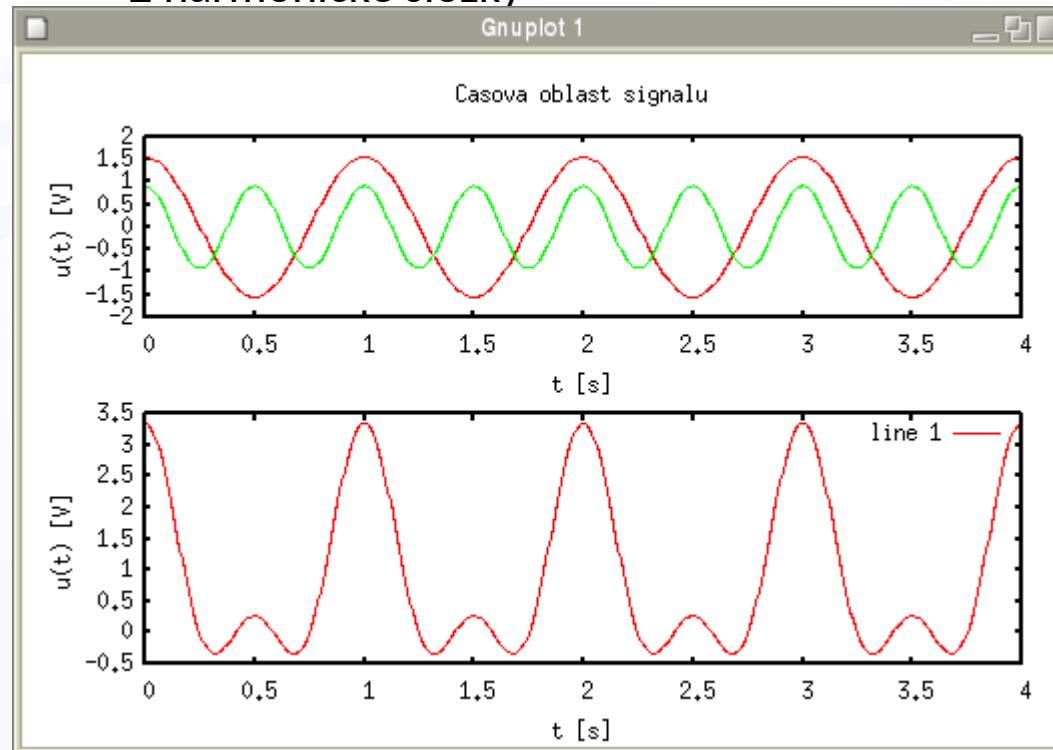
$$a_n = \frac{2}{T} \int_0^T f(t) \cos (n\omega_0 t) dt \quad b_n = \frac{2}{T} \int_0^T f(t) \sin (n\omega_0 t) dt$$



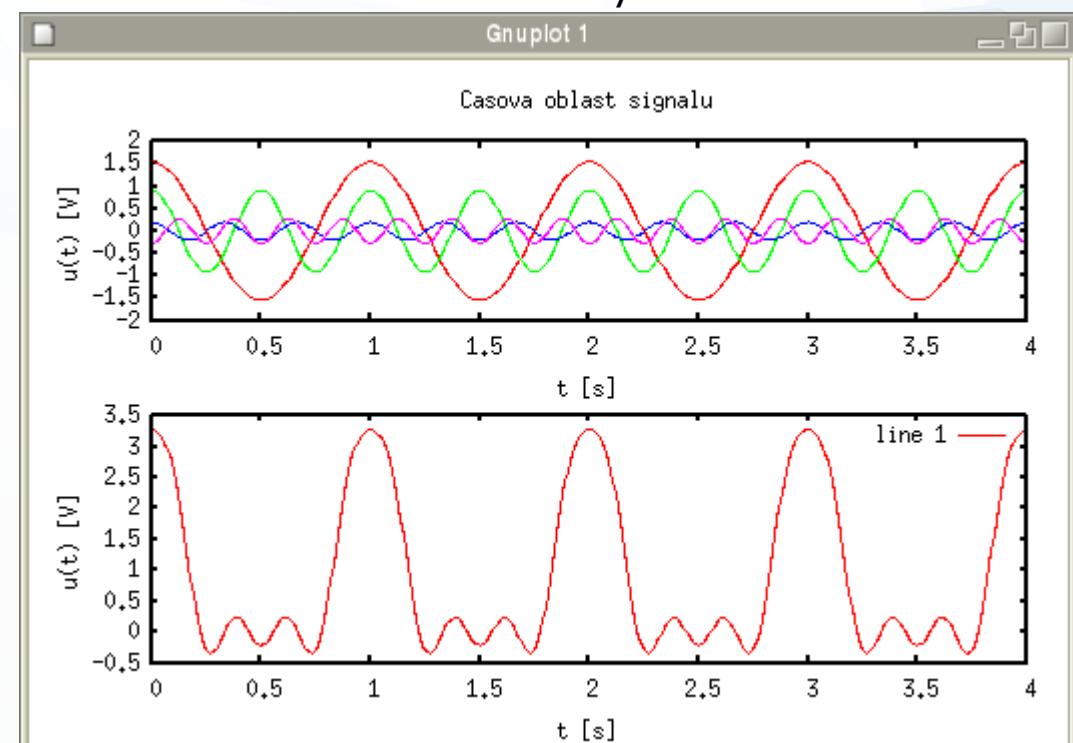
# L1: Fourierův rozklad

- Umožňuje libovolný signál transformovat na součet harmonických signálů, které jsou celým násobkem základního signálu

2 harmonické složky

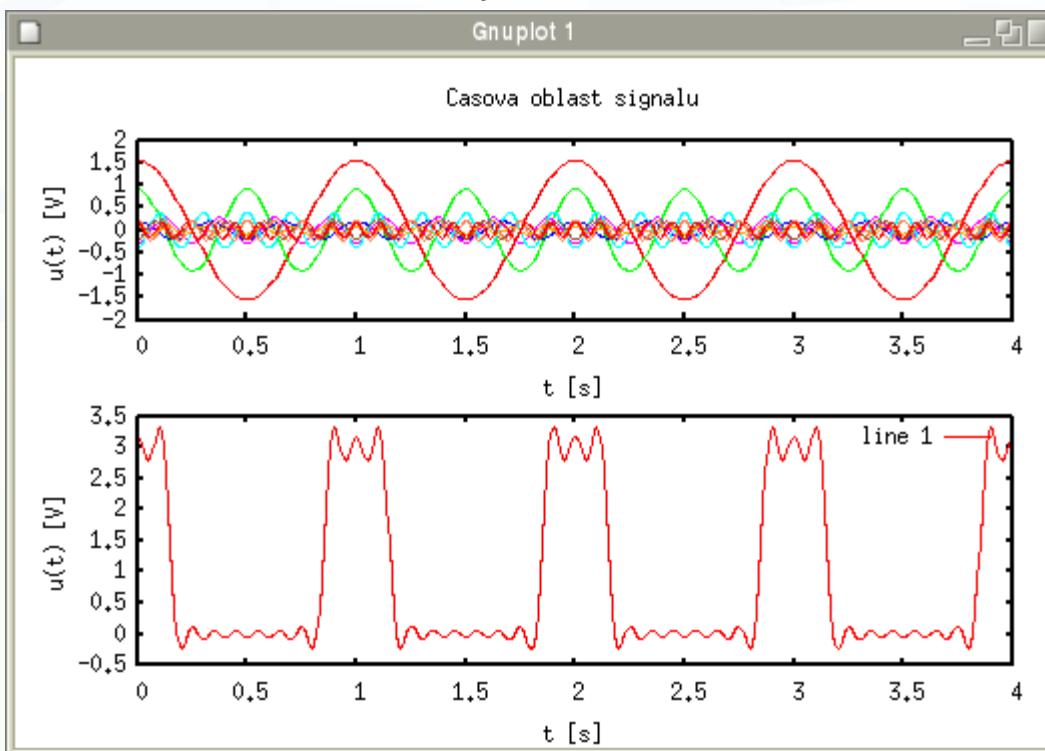


4 harmonické složky

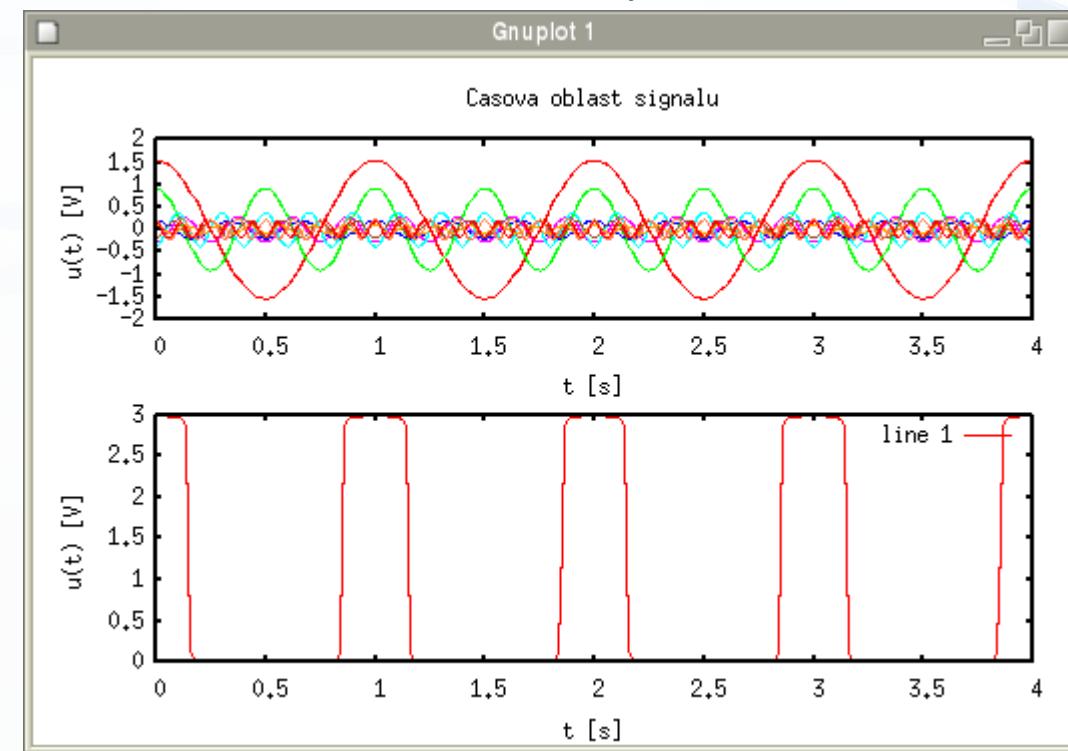


# L1: Fourierův rozklad - pokračování

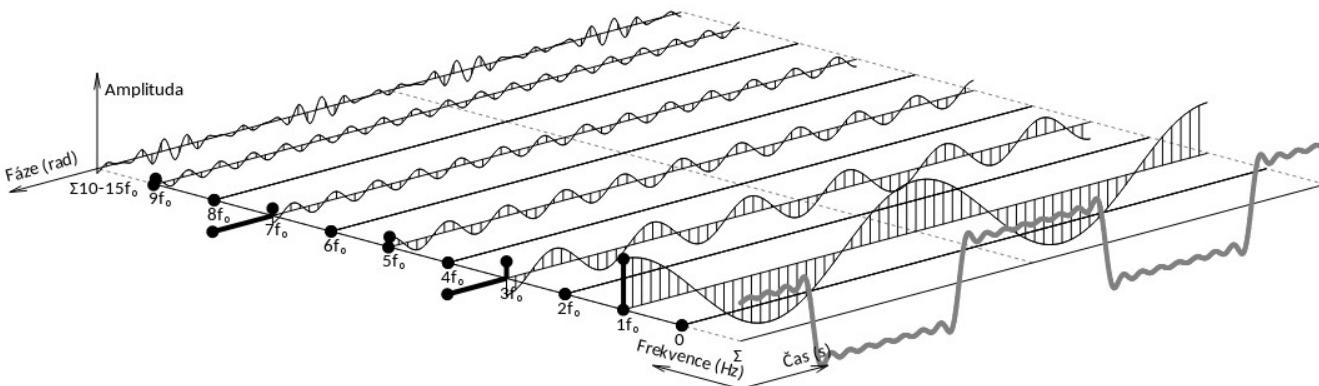
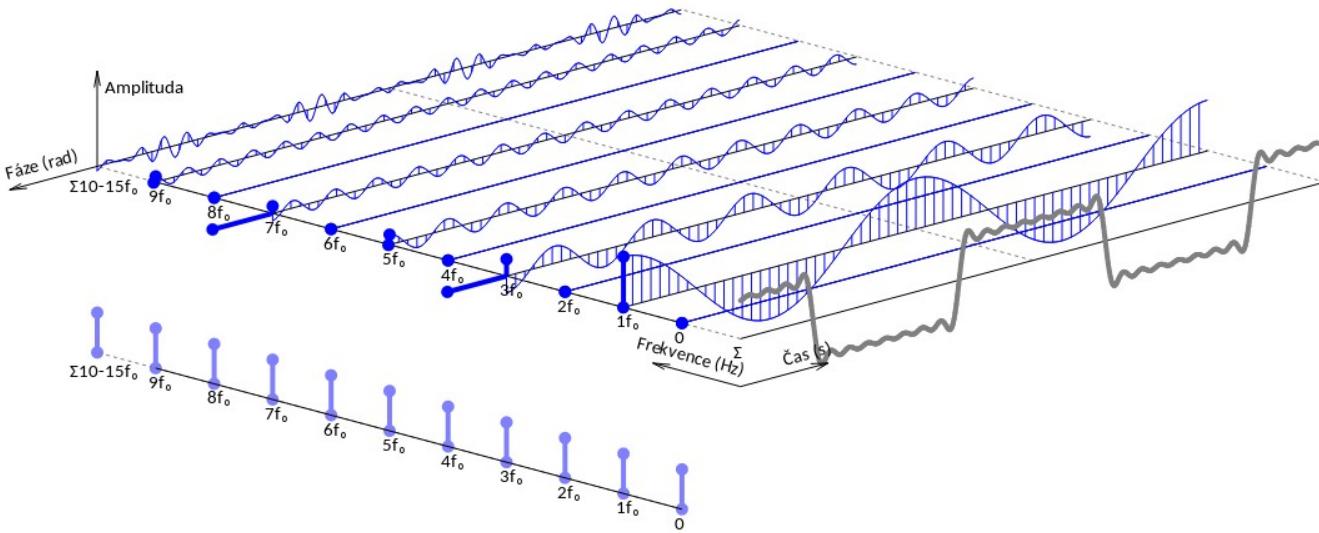
10 harmonických složek



100 harmonických složek



# L1: Fourierův rozklad - pokračování



zdroj: <https://tomasboril.cz/fourierseries3d/cz/>

# L1: Limitace šířkou pásma

- Šířka pásma
  - Rozmezí frekvencí, které je vodič schopen přenést
  - Signály s frekvencí mimo definovanou šířku pásma se nepřenesou vůbec
    - Tedy například v Fourierově transformaci musíme přenášet méně přesný signál než bychom chtěli
  - Čím větší šířka pásma tím více harmonických signálů můžu přenášet
  - Čím na vyšší frekvence první harmonické, tím vyšší přenosová rychlosť, ale nižší počet dalších harmonických

Bps	T (msec)	First harmonic (Hz)	# Harmonics sent
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

# L1: Modulační – baudová rychlosť ( Nyquistovo kritérium )

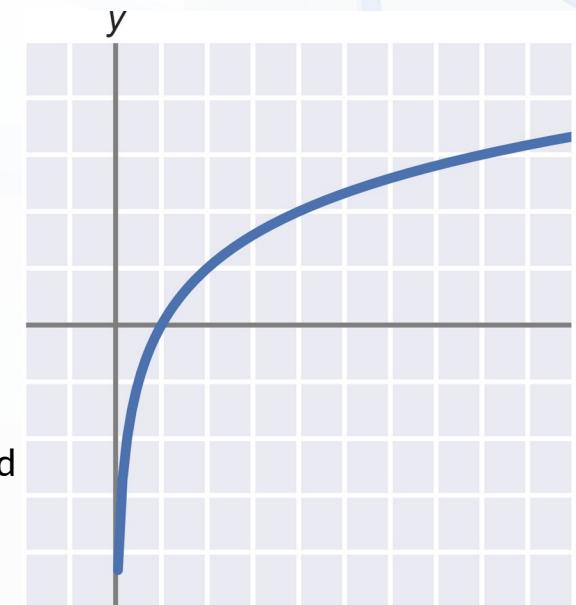
- Udává jak rychle jsme schopni měnit stavy/parametry signálu
- Je logicky závislá na šířce pásma
- Její hodnotu určuje Nyquistovo kritérium, které říká:
  - $v_m = 2 * W$
  - $v_m$  – modulační / baudová rychlosť, jednotka jeden Baud [Bd]
  - W – šířka pásma, tedy rozdíl mezi nejvyšší a nejnižší přenositelnou rychlosťí

# L1: Modulační – baudová rychlosť - příklad

- Máme přenosový kanál, který umožňuje přenášet frekvence od 100kHz do 1MHz, jaká je modulační rychlosť ?
  - $W = 1.000.000 - 100.000 = 900.000 \text{ Hz}$
  - $v_m = 2 * W = 2 * 900.000 = 1.800.000 \text{ Bd}$
- Máme přenosový kanál, který umožňuje přenášet frekvence od 900kHz do 1MHz, jaká je modulační rychlosť ?
  - $W = 1.000.000 - 900.000 = 100.000 \text{ Hz}$
  - $v_m = 2 * W = 2 * 100.000 = 200.000 \text{ Bd}$
- Čím větší je rozsah frekvencí ( šířka pásma ), kterou je přenosový kanál schopen přenést, tím větší je modulační rychlosť.
- **!!! POZOR – změna modulační rychlosti neříká sama přímo nic o změně rychlosti přenosové !!!**

# L1: Přenosová rychlosť / kapacita přenosového kanálu

- Uvažujme **Přenosový Kanál Bez Šumu**
  - Pro zjednodušení – v realitě neexistuje
- Vyjdeme z Nyquistova kritéria – čím vyšší modulační rychlosť, tím vyšší přenosová rychlosť
  - Logicky čím rychleji můžu měnit stav signálu, tím více dat za jednotku času přenesu
- Zároveň je ale třeba brát v potaz, že signál nemusí být binární, ale může přenášet více stavů, které můžeme reprezentovat jako 0,1,11,00,01,10 a tím přenosovou rychlosť opět zvýšit
- Obecně platí  $v_p = 2 * W * \log_2(V)$  [ b/s ],  $v_p = v_m * \log_2(V)$ 
  - $W$  – šířka pásma [ Hz ]
  - $V$  – počet stavů
  - Čím větší je modulační rychlosť nebo čím vyšší je počet stavů, tím vyšší je přenosová rychlosť
    - Modulační rychlosť je daná šířkou pásma a ta typicky nejde snadno měnit
    - Počet stavů můžeme přidávat snáze, ale ne neomezeně, protože musíme být stále schopni dva stavů od sebe bezpečně rozoznat



# L1: Přenosová rychlosť / kapacita prenosového kanálu - príklad

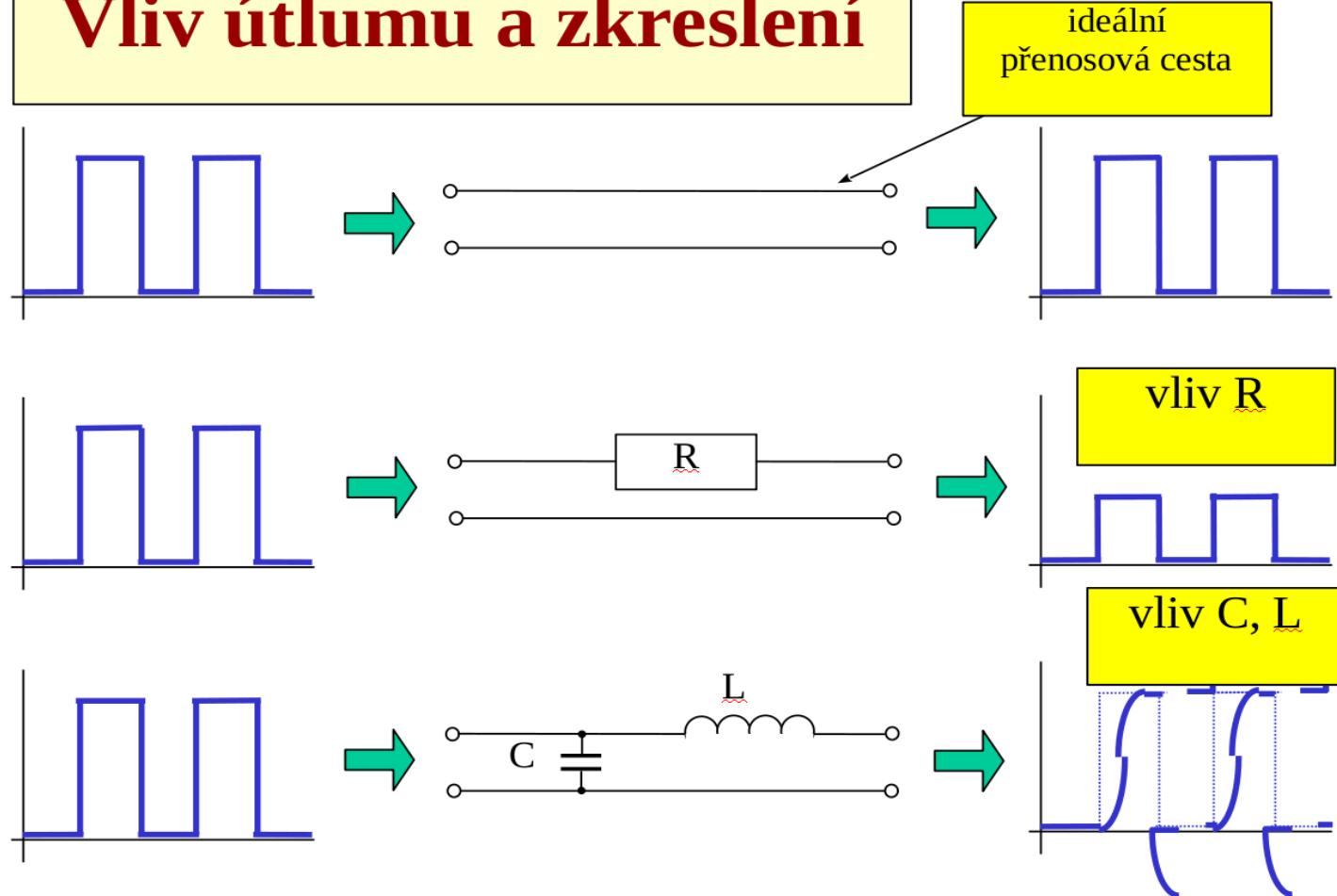
- Máme prenosový kanál bez šumu, ktorý dokáže prenášet frekvencie od 100Hz 1000Hz a v ktorém je možné identifikovať 4 úrovne signálu. Jaká je kapacita tohto prenosového kanálu ?
  - $v_p = 2 * W * \log_2(V)$  [ b/s ]
  - $W = 1000 - 100 = 900$  Hz
  - $V = 4$
  - $v_p = 2 * W * \log_2(V) = 2 * 900 * \log_2(4) = 2 * 900 * 2 = 3.600$  b/s = 3.6 kb/s = 3.6 kbps
- Jak sa změní kapacita prenosového kanálu z predchozího príkladu, pokud se počet stavu zvýší o 4 ?
  - $v_p = 2 * W * \log_2(V)$  [ b/s ]
  - $W = 1000 - 100 = 900$  Hz
  - $V = 4 + 4 = 8$
  - $v_p = 2 * W * \log_2(V) = 2 * 900 * \log_2(8) = 2 * 900 * 3 = 5.400$  b/s = 5.4 kb/s = 5.4 kbps
  - Změna =  $5400 / 3600 = 1.5$ , prenosová rychlosť se zvýší 1.5x

# L1: Útlum, zkreslení a rušení

- V ideálním světě je přenos ovlivňován
  - Útlum
    - Vlivem vzdálenost – délky přenosového kanálu dochází ke snížení intenzity/kvality signálu
    - Vlivem prvků v zařízeních, kterými signál prochází dochází také k útlumu – například na odporu
  - Zkreslení
    - Vlivem prvků v zařízeních, kterými signál prochází dochází také k útlumu – například cívka či kondenzátor
  - Rušení
    - Vnější – může jít například o silové vedení v blízkosti nestíněného vodiče
      - Nemusí se projevovat konstantně, ale může se v čase měnit podle spotřeby
    - Vnitřní – i jednotlivé vodiče například v twistu se mohou ovlivňovat
      - Defakto každý vodič vedoucí elektřinu se zároveň chová jako anténa i vysílač
      - Proto se páry v twisty kříží aby se rušení eliminovalo
      - Další možností je použití stíněných vodičů ( uvnitř i vně vodiče – stínění mohou mít jednotlivé vodiče, páry i celý svazek )
- Čím delší je datové cesta – vzdálenost, kterou musejí data urazit - tím se tyto negativní vlivy projevují více

# L1: Rušení a útlum - příklady

## Vliv útlumu a zkreslení



# L1: Shannonovo kritérium

- V ideálním světě je přenos ovlivňován
  - Rušení, útlum, zkreslení ...
- Kapacita přenosového kanálu bude ovlivněna kvalitou daného kanálu
  - Tedy jak věrně/bezpečně dokáže data přenést a tím i jak lze rozlišit jednotlivé stavy
- Pro reálný přenosový kanál platí Shannonovo kritérium
  - $v_p = W * \log_2(1+S/N)$  [b/s]
  - W - šířka pásma [ Hz ]
  - S - úroveň signálu
  - N - úroveň šumu
- V reálu je tedy zásadnější kvalita signálu a přenosového media a tedy nízké hodnota šumu / rušení
  - Signál je třeba zesilovat
  - Vodič je třeba odstínit od okolí, ale i od ostatních žil v rámci daného vodiče - například v twistu

# L1: Shannonovo kritérium - příklad

- Mějme přenosový kanál, ve kterém je poměr signálu a šumu 1000:1 a šířka pásma, kterou kanál dokáže přenášet je 3.1KHz ( telefonní linka v optimálním případě ). Jaká je kapacita takového kanálu?
  - $v_p = W * \log_2(1+S/N)$  [b/s]
  - $W = 3100$  Hz
  - $S= 1000, N=1 \Rightarrow S/N = 1000/1 = 1000$
  - $v_p = 3100 * \log_2(1+1000) = 3100 * 9,97 = 30.907$  b/s = 30,907 kbps
- Jak se změní kapacita kanálu, pokud se úroveň šumu 1x zvýší ?
  - $v_p = W * \log_2(1+S/N)$  [b/s]
  - $W = 3100$  Hz
  - $S= 1000, N = 2 \Rightarrow S/N = 1000/2 = 500$
  - $v_p = 3100 * \log_2(1+500) = 3100 * 8,97 = 27.807$  b/s = 27,807 kbps
  - **Změna = 27.807/31.907 = 0,89, přenosová rychlosť klesne o 11%**

# L1: Zesílení a decibely

- Decibel je logaritmická jednotka určující poměr dvou veličin
  - Základem není Bel ale DeciBel - ve vzorci je třeba násobit výsledek deseti
  - Může sloužit k popisu změny - například signálu, kde můžeme měřit zesílení nebo zeslabení např výkonu, proudu nebo napětí - označujeme jako  $A_x$  - kde x specifikuje zdroj poměru
  - **Výkonový:**  $A_p = 10 \cdot \log_{10}(P_{out}/P_{in})$  [dB]
    - ( 10x právě proto, že výsledek je v deciBelech )
  - **Proudový:**  $A_i = 20 \cdot \log_{10}(I_{out}/I_{in})$  [dB]
    - ( 20x proto, výkon je úměrný kvadrátu intenzity pole - případně napětí či proudu)
  - **Napěťový:**  $A_u = 20 \cdot \log_{10}(U_{out}/U_{in})$  [dB]
    - ( 20x proto, výkon je úměrný kvadrátu intenzity pole - případně napětí či proudu)

# L1: Zesílení a decibely příklad

- Jaké je bude zesílení/zeslabení pokud vstupní výkon bude 10mW a výstupní bude 5mW ?
  - $A_p = 10 * \log_{10}(P_{out}/P_{in})$
  - $P_{in} = 10 \text{ mW}$
  - $P_{out} = 5 \text{ mW}$
  - $A_p = 10 * \log_{10}(5/10) = 10 * -0,30 = -3\text{dB} \Rightarrow \text{dojde k útlumu o } 3\text{dB} (\sim \text{na polovinu})$
- Jaké je bude zesílení/zeslabení pokud vstupní výkon bude 5mW a výstupní bude 15mW ?
  - $A_p = 10 * \log_{10}(P_{out}/P_{in})$
  - $P_{in} = 5 \text{ mW}$
  - $P_{out} = 15 \text{ mW}$
  - $A_p = 10 * \log_{10}(15/5) = 10 * 0,48 = 4,8\text{dB} \Rightarrow \text{dojde k zesílení signálu o } 4,8\text{dB}$

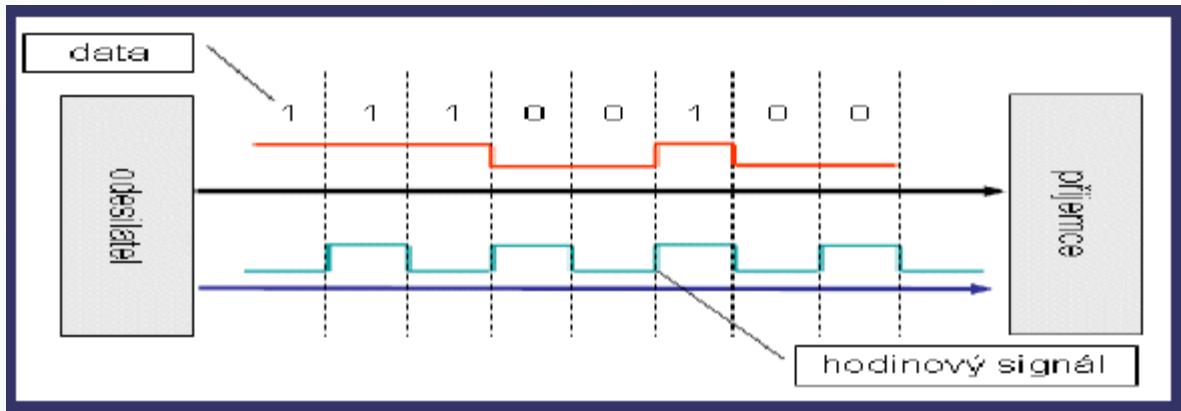
# L1: Shannonovo kriterium a zeslabení signálu - příklad

- Jaké je bude kapacita kanálu, pokud šířka pásma bude 3.1kHz a zesílení/zeslabení signálu bude 30dB ?
  - $A_p = 10 * \log_{10} (P_{out}/P_{in})$ ,  $v_p = W * \log_2 (1+S/N)$  [b/s]
  - Poměr signálu a šumu se rovná poměru vstupního a výstupního výkonu signálu
    - $S/N = P_{out}/P_{in} = 10^{(A_p/10)} = 10^3 = 1000$ 
      - Protože  $y = \log_a(x) \iff a^y = x$
    - $v_p = W * \log_2 (1+S/N) = 3100 * \log_2 (1+1000) = 3100 * 9,97 = 30,907 \text{ kbps (kb / s)}$
  - Poměr signálu a šumu tedy můžeme udávat i ve formě dB

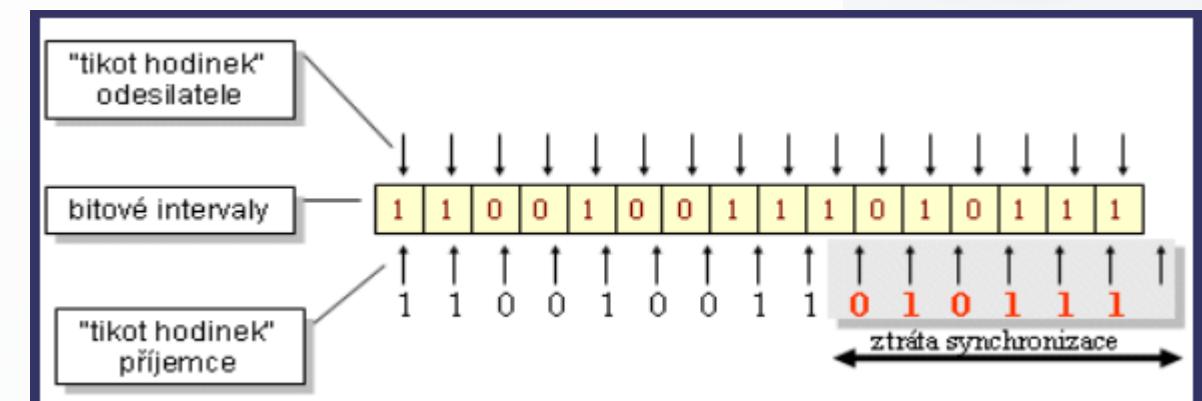
# L1: Synchronizace

- Signál může mít jeden či více stavům které od sebe potřebuje přesně rozoznávat
- Data ( 0,1 ) jsou posílána signálem v „pulzech“ - každý pulz přenáší jeden stav
- Pulz – tik hodin – musí mít konstantní délku a musí jí znát vysílač i přijímač
  - Logicky, jinak bychom více stavů mohli interpretovat jako jeden či naopak jeden stav jako více stavů

Ideální přenos – odesílatel a příjemce jsou synchronní

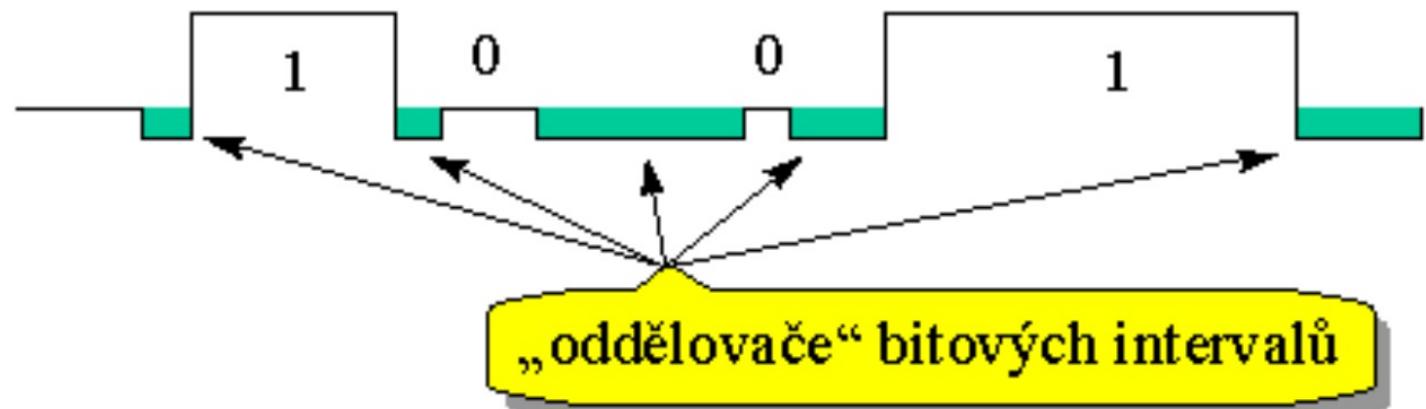


Problém v desátém pulzu



# L1: Synchoronizace: Asynchronní přenos

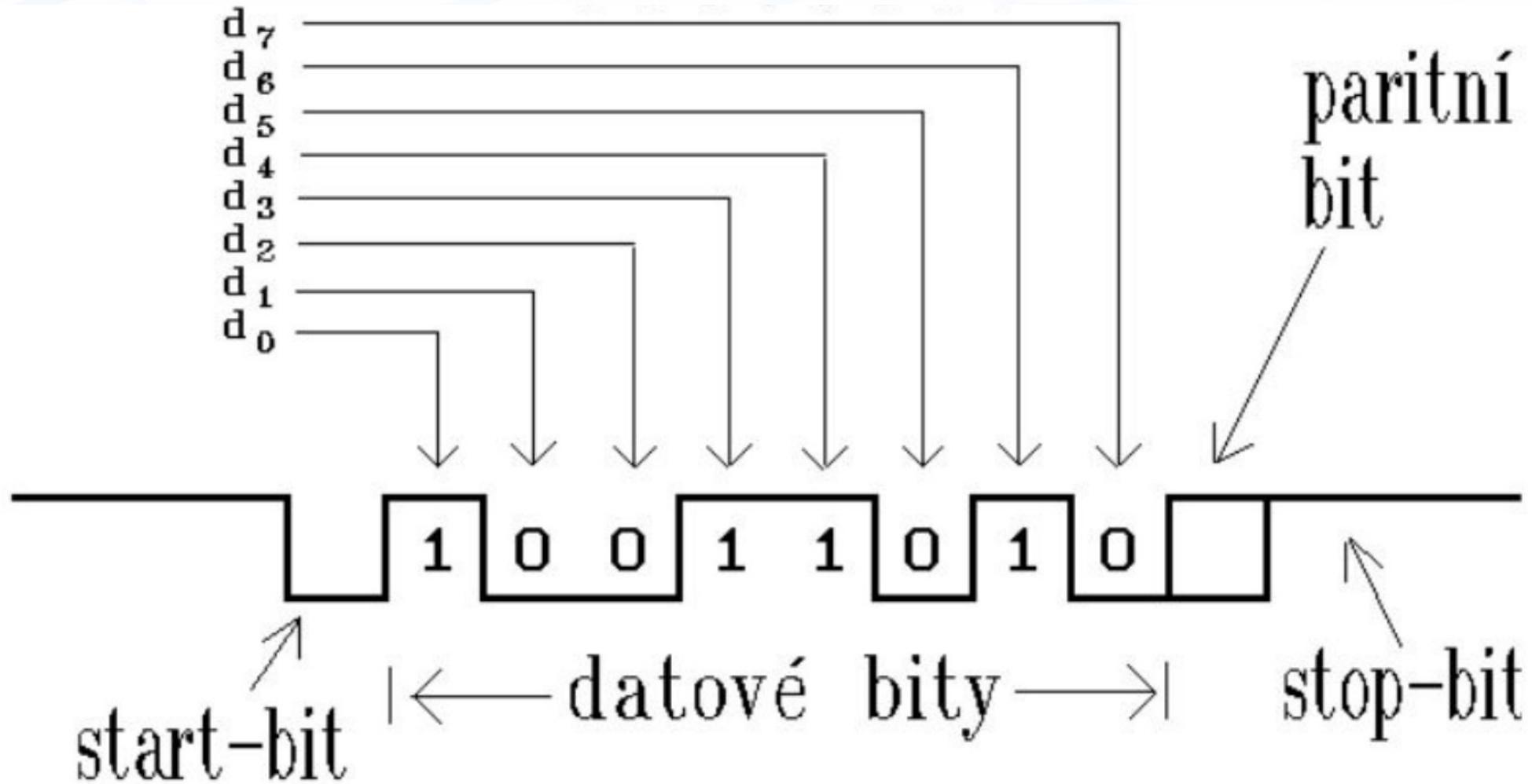
- Zde jednoduše žádná trvalá synchronizace není
- **Vysílač může vysílat v různé časy – neřídí se hodinami ...**
- **Délka pulzu se může pro jednotlivé pulzy lišit ....**
- Může to vůbec fungovat ?!
- Může, ale je nákladné, protože to vyžaduje **tří stavovou logiku** – min tři stavy
  - Dva stavy na přenos požadovaných dat 0 a 1
  - Třetí stav pro informaci, zda se začíná/končí přenos dat
- Výhodou je, že nepožaduje žádnou synchronizaci
- V praxi se příliš nepoužívá



# L1: Synchronizace: Arytmický přenos

- „Speciální“ varianta asynchronního přenosu
- **Vysílač může vysílat v různé časy – neřídí se hodinami ...**
- **Délka pulzu ( jednoho bitu ) je pevně daná**
- Vycházíme z předpokladu, že trvalou synchronizaci se udržet nepovede, ale na omezenou dobu ano
  - Jinak řečeno na dlouhé přenosy se čas rozjede, ale na krátké jej dokážeme udržet „synchronní“
    - Možná ne dokonale, ale tak aby se neprekročila hranice bitů a to stačí
- Potřebujeme nějak zajistit prvotní synchronizaci
  - To by zas šlo extra vodičem, ale to jsme u synchronní varianty
- Doplníme signál o „start“ a „stop“ značky/bity
  - Pozor nejdříve se o jeden bit – jen se to tak nazývá !!!
- Princip fungování
  - Posloucháme komunikaci a neděláme nic, dokud nepřijde „start bit“
  - Od přijetí start bitu začnu s dohodnutou frekvencí vzorkovat data – rozpoznávat jednotlivé bity
    - Úvodní synchronizaci nám zajistil start bit a předpokládáme, že během krátkého přenosu se hodiny udrží synchronní
    - Pokud nastane i přesto chyba, na L1 to nepoznáme, ale zjistí to L2 – řekneme si později
  - Počet dat může být pevně stanoven a nebo může být dohodnuta ukončovací značka – stop bit

# L1: Synchronizace: Arytmický přenos příklad



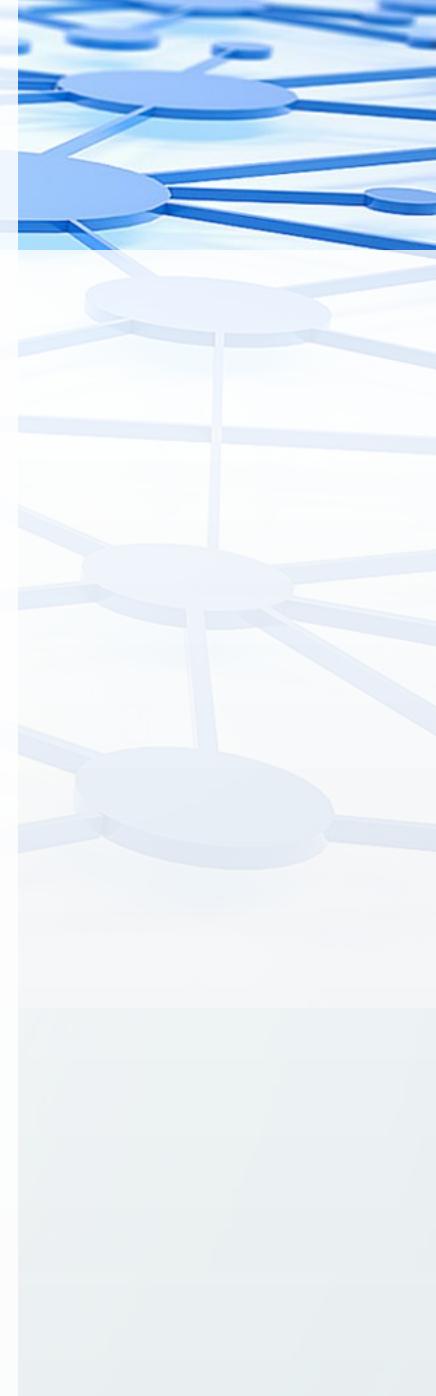
# L1: Synchronizace: Arytmický přenos III.

## Paritní bit

- Paritní bit slouží k základnímu ověření správnosti přenosu
- Paritní bit doplňuje bitvou posloupnost o jeden bit, který doplňuje počet jedniček v posloupnosti na lichý nebo sudý počet
- Parity tedy mohou v základ být dvě
  - Sudá parita => **1101|1** ( datové jedničky jsou tři, aby jich byl sudý počet musím jednu doplnit )
  - Lichá parita => **1101|0** ( datové jedničky jsou tři, což je liché, tedy v paritním bitu nedoplňuji nic )
- Jednoduchá implementace, protože se jedná jen o jednoduchý čítač počtu
- Nemusí odhalit násobnou chybu
  - Přenáším se sudou paritou: **1101|1** ( doplnil jsem 1 na sudý počet )
  - Dorazilo mi: **1000|1** - výsledná parita parita je 1 a tedy souhlasí, ALE v datech jsou dvě chyby
  - Stejně tak nepozná, zda chyba byla jen jedna nebo více
  - Důkladnější zabezpečení se realizuje až na L2 – řekneme si následně

# L1: Synchronizace: Arytmický přenos IV.

## Využití kapacity přenosového kanálu



- U arytmického přenosu jsme se poprvé potkali s tím, že kromě dat, která chceme přenášet, přenášíme navíc i data „řídící“
- Tyto data „navíc“ my vlastně přenášet nechceme, ale musíme aby se přenos realizoval
- Jedná se o
  - Start bit
  - Stop bit
  - Paritní bit
- Tato data nám „snižující“ kapacitu přenosového kanálu – opticky nám snižují přenosovou rychlosť, protože „zabírají“ místo informačním datům
- Využití kapacity přenosové kanálu může spočítat jako :
  - $n = N/M * 100 [\%]$ 
    - N – počet datových bitů – ty co opravdu chceme přenášet
    - M – počet všech bitů, které musíme přenášet

# L1: Synchronizace: Arytmický přenos IV.

## Využití kapacity přenového kanálu příklad

- Příklad:

Mějme arytmický přenos s délkou informačních dat 8 bitů, s 1 start bitem, jedním paritním bitem a dvěma stop bity. Jaké je využití kapacity přenosového kanálu ?

$$n = N/M$$

$N = 8$  bitů ( ty chceme přenášet )

$M = 8 + 1 + 1 + 2 = 12$  ( datový byty + start bit + paritní bit + dva stop bity )

$$n = 8/12 = 0,6667 = 66,67\%$$

Tedy náš arytmický přenos spotřebuje 1/3 kapacity přenosového kanálu.

# L1: Synchronizace: Arytmický přenos V.

## Využití kapacity přenového kanálu versus délka přenosu

- Z příkladu využití kapacity přenosového kanálu je jasné, že bychom rádi co nejdelší přenos, aby využití kanálu bylo co nejlepší
  - Pokud budeme mít 1 start bit, 1 paritní bit a jeden stop bit a délku dat **8bit**
    - Využití kapacity kanálu bude  $n = N/M$ ,  $N = 8$  b,  $M = 8 + 1 + 1 + 1 = 11$  b,  $N = 8/11 = 72,72\%$
  - Pokud budeme mít 1 start bit, 1 paritní bit a jeden stop bit a délku dat **16bit**
    - Využití kapacity kanálu bude  $n = N/M$ ,  $N = 16$  b,  $M = 16 + 1 + 1 + 1 = 19$  b,  $N = 16/19 = 84,42\%$
  - Čím delší budou přenášená data, tím bude vyšší využití kapacity přenosového kanálů
  - ALE zároveň čím delší budou přenášená data tím vyšší bude pravděpodobnost chyby/ztráty synchronizace v rámci arytmického přenosu
  - Chceme tedy co nejdelší datovou část, ale NE delší než by došlo ke zvýšení pravděpodobnosti chyby nad námi stanovenou mez
  - Pravděpodobnost úspěšného přenosu  $N$  bitů:  $P_N = p_1^N$ 
    - $p_1$  - pravděpodobnost úspěšného přenosu 1 bitu
    - $q_1$  - pravděpodobnost chyby v přenosu 1 bitu,  $q_1 = 1-p_1$

# L1: Synchronizace: Arytmický přenos V.

## Využití kapacity přenového kanálu versus délka přenosu příklad

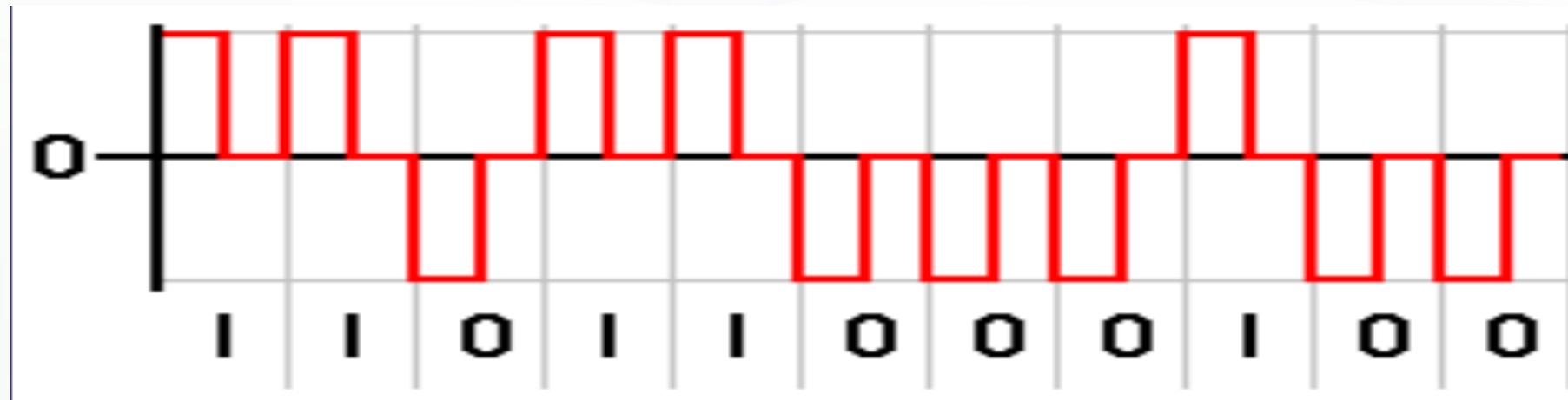
- Mějme Symetrický Binární Přenosový Kanál Bez Paměti, kolik můžeme přenést bitů v rámci jediného přenosu, aby pravděpodobnost bezchybného přenosu neklesla pod 99%, pokud pravděpodobnost bezchybného přenosu 1 bitu je 99.99%
  - $P_N = p_1^N$
  - $P_N = 99\% = 0,99, p_1 = 99,99\% = 0,9999, N = ?$
  - $N = \log_{p_1}(P_N) = \log_{0,9999}(0,99) = 100,4983 = \mathbf{100b} \text{ ( nemůžeme jít nad 100b )}$
- Pokud budeme chtít mít pravděpodobnost bezchybného přenosu neklesla pod 99,9, kolik bitů můžeme přenést ?
  - $P_N = p_1^N$
  - $P_N = 99,9\% = 0,999, p_1 = 99,99\% = 0,9999, N = ?$
  - $N = \log_{p_1}(P_N) = \log_{0,999}(0,99) = 10,0045 = \mathbf{10b} \text{ ( nemůžeme jít nad 10b )}$

# L1: Synchronizace: Synchronní přenos

- Ideální stav
- Odesílatel a příjemce mají „nějak“ zajištěnou trvalou synchronizaci hodin
  - POZOR nejedná se o reálný čas NTP – jde o délku a start pulzu
- Pro zajištění trvale synchronizace může být použit extra vodič s hodinovým pulzem
  - Bude těžko / drahou realizovatelné na velké vzdálenosti
- Může modifikovat signál tak, aby v sobě kromě požadovaného rozlišení bitů s hodnotou 0 a 1 přenášel i informaci o hodinách
  - Může se jednat o místo jde je možné se sesynchronizovat ( NRZ )
  - Může se jednat přímo o signál, který v sobě nese hodiny ( RZ, Manchester )
- Nedochází k žádné změně ani doplnění dat – jde „pouze“ o změnu toho, jakým, signálem jsou data přenášena
- Tomuto způsobu zajištění synchronizace se říká „kódování“

# L1: Kódování signálu: RZ

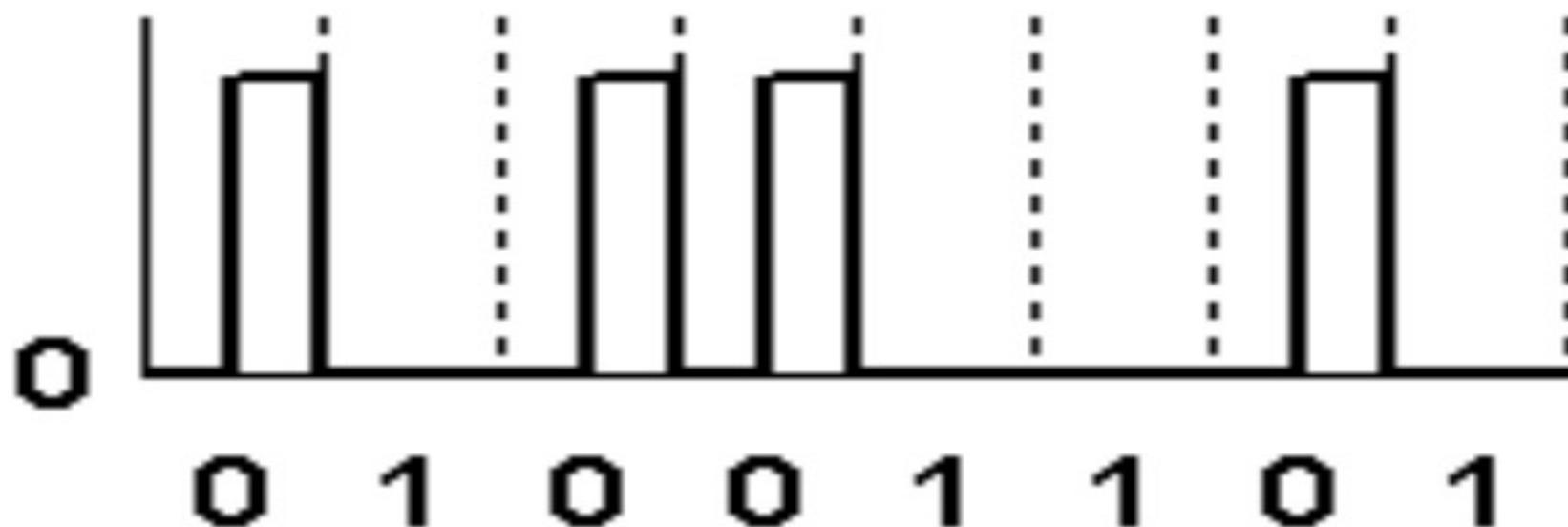
- RZ - Return to Zero
- Patrně nejjednodušší kódování
- 0 a 1 jsou reprezentovány kladnými a zápornými pulzy
- Třetí stav je „nulový“ nebo „neutrální“ a k němu se signál po každém pulzu vrací
- „Nese“ v sobě hodiny, neboť ke změně dochází v každém pulzu bez ohledu na data



zdroj: <https://cs.wikipedia.org>

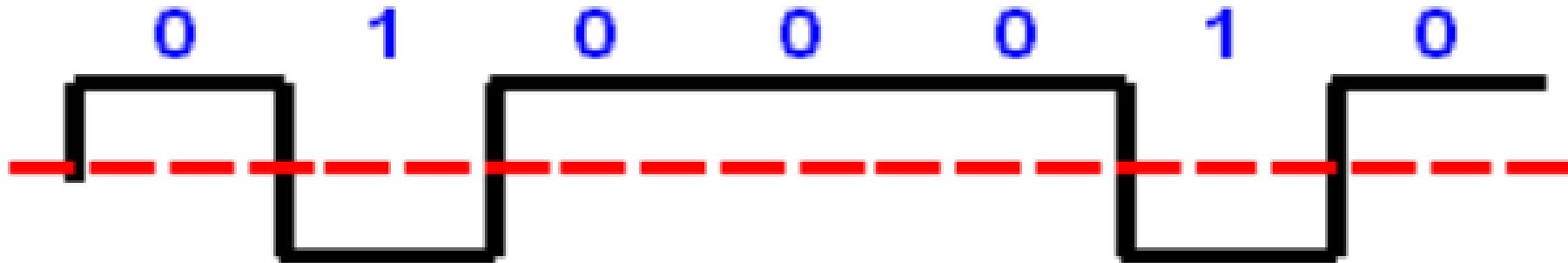
# L1: Kódování signálu: RZI

- RZI – Return To Zero Inverted
- Používá už jen dvě úrovně
  - 0 – kratší změna signálu než je délka hodin ( ale je tam nějaký pulz )
  - 1 – bez změny signálu
- Hodiny u v sobě přímo nemá – například pro posloupnost jedniček nedochází ke změně signálu
- **Pokud v signálu je 0 je možné hodiny synchronizovat na konci pulzu**
- **Odolný vůči změně polarity**



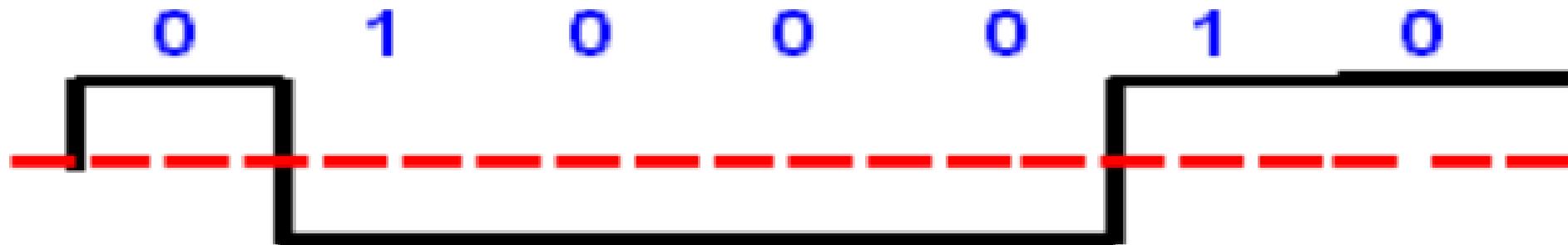
# L1: Kódování signálu: NRZ

- NRZ – Non Return To Zero
- Používá jen dva stavy
  - 0 - například záporný pulz
  - 1 - například kladný pulz
- Mezi jednotlivými stejnými bity ( více nul nebo více jedniček ) nedochází ke změně
- **Není možné použít k synchronizaci hodin** a je třeba řešit synchronizaci jinak - externě



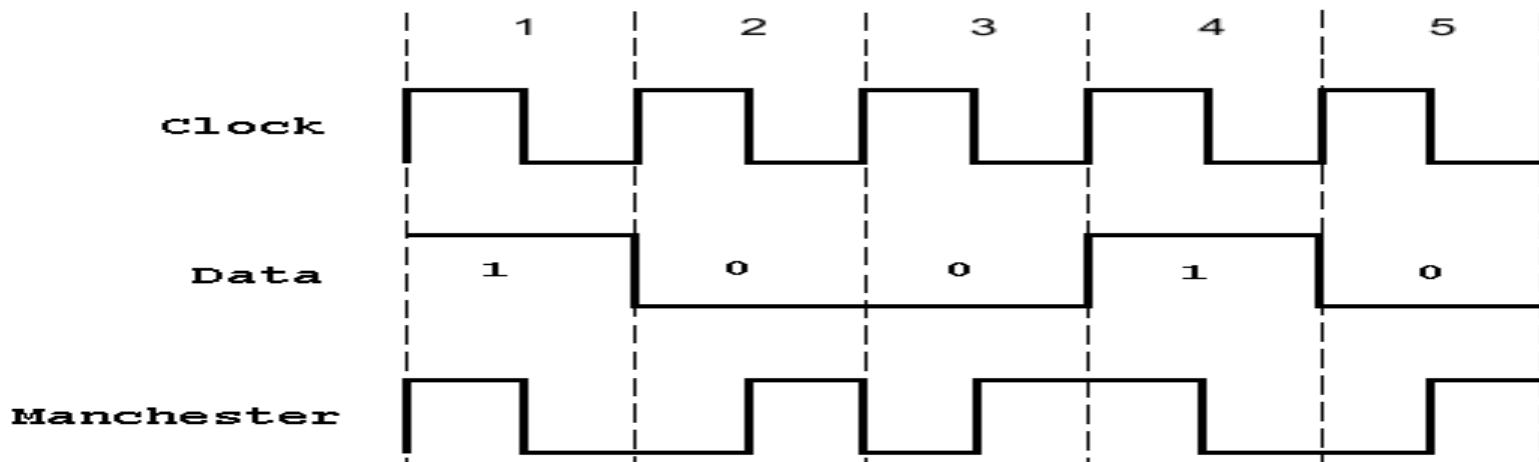
# L1: Kódování signálu: NRZI

- NZRI – Non Return To Zero Inverted
- Má jen dvě úrovně signálu, ale ty přímo nesignalizují 0 a 1
- Bity 0 a 1 jsou rozlišeny:
  - 0 – nenastává změna v signálu ( je jedno zda je právě v kladné či záporné hodnotě signálu )
  - 1 – nastává změna stavu na začátku pulzu
- **Může sloužit k synchronizaci hodin** – s každou jedničkou vím kde jsou hodiny
- **Je odolný vůči změně polarity** – hodnoty nejsou definované konkrétním stavem, ale změnou či zachováním stavu



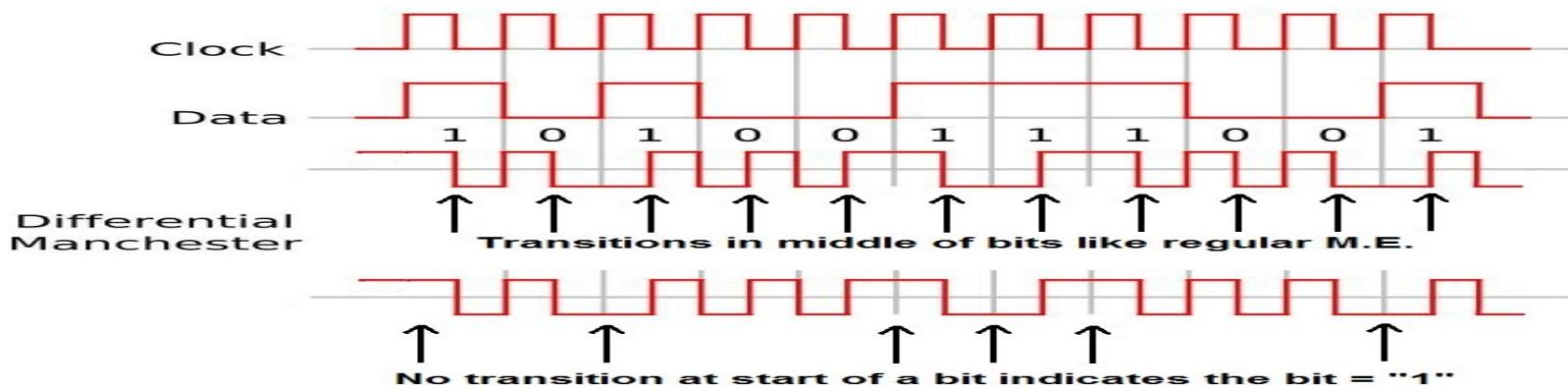
# L1: Kódování signálu: Manchester

- Manchester má opět jen dva stavů
- Rozdělení nuly a jedničky se odečítá uprostřed pulzu, kde VŽDY musí nastat změna
  - **Můžeme dobře synchronizovat hodiny**
- Hodnoty ( například, může být i otočená varianta )
  - 0 – změna signálu z kladného na záporný
  - 1 – změna pulzu ze záporného na kladný

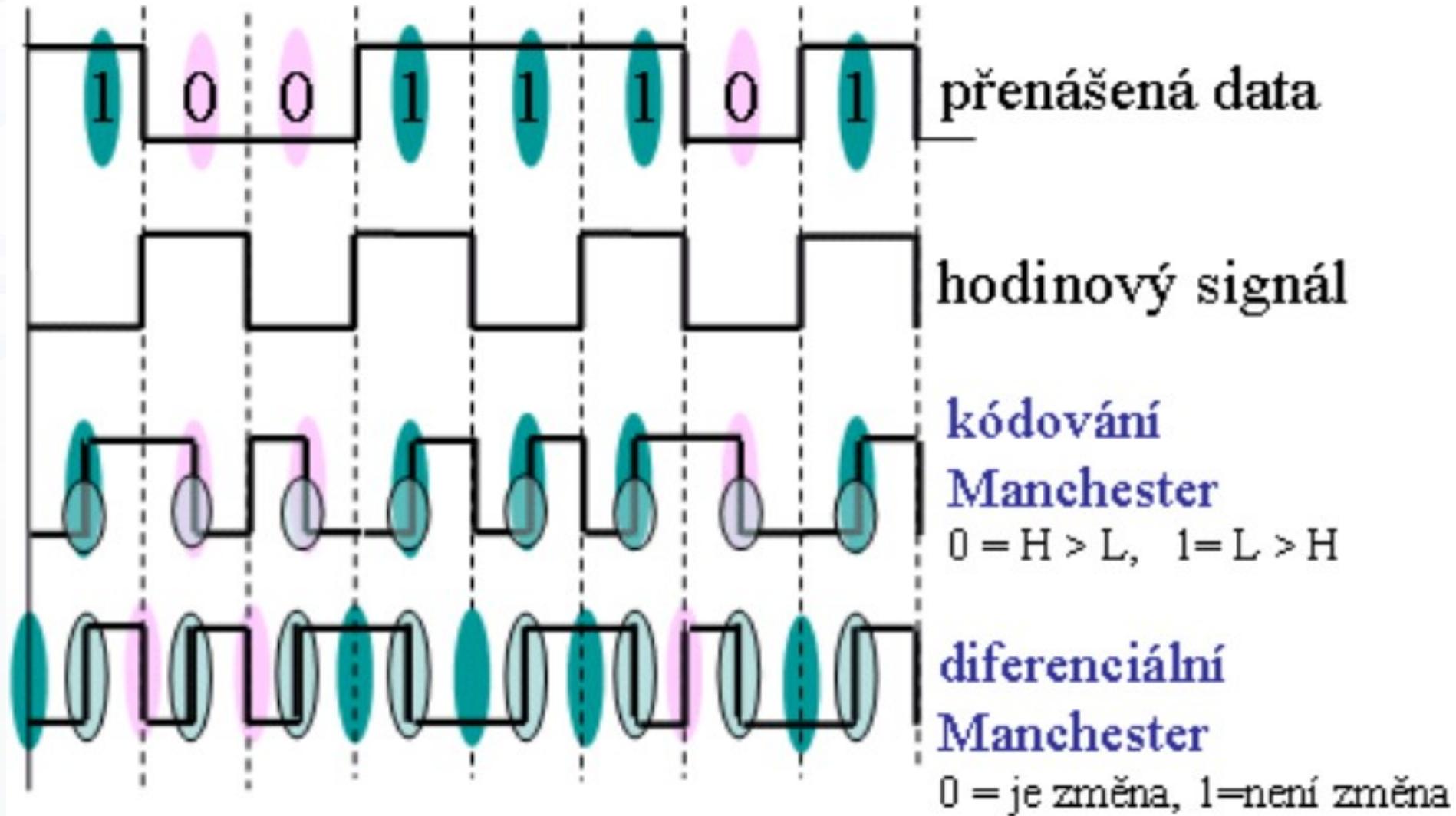


# L1: Kódování signálu: Diferenciální Manchester

- Diferenciální Manchester opravu nedostatky klasické verze
- Opět má jen dva stavů a vždy uprostřed pulzu dojde ke změně stavu ( nezávisle na datech ) - **přímo v sobě nese hodiny.**
- Data jsou odečítána na vstupní hraně pulzu a to následovně:
  - 0 – na vstupní hraně pulzu nastala změna ( nezáleží na směru )
  - 1 - na vstupní hraně pulzu nenastala změna
- **Je odolný vůči změně polarity**



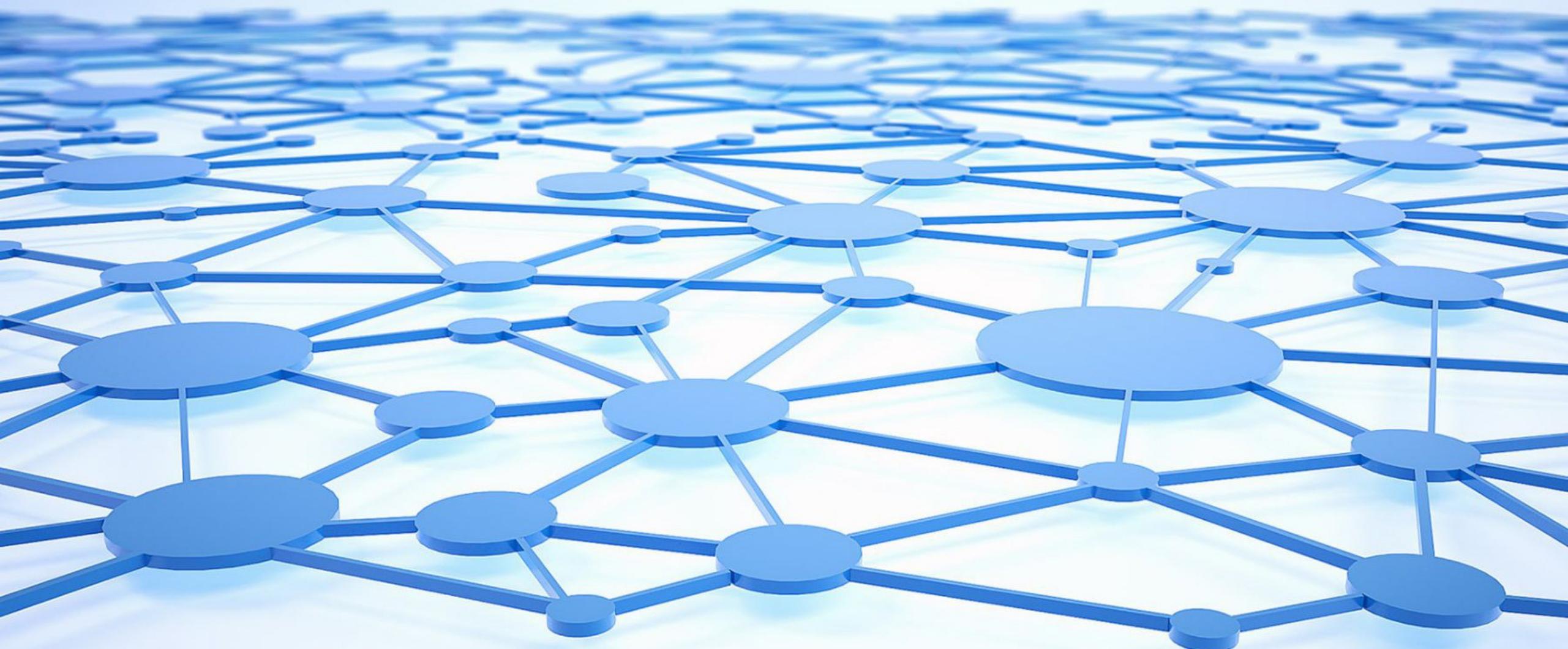
# L1: Kódování signálu: Klasický a Diferenciální Manchester



# Úvod do počítačových sítí

Přednáška 4 ( 2025/2026 )

ver. 2025-10-01-01

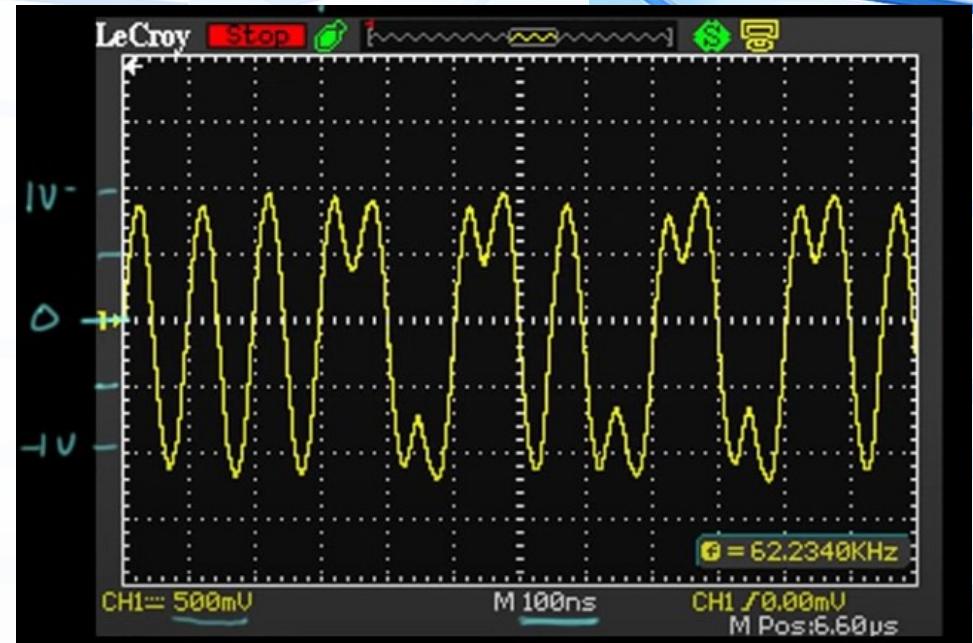


# L1: Opakování

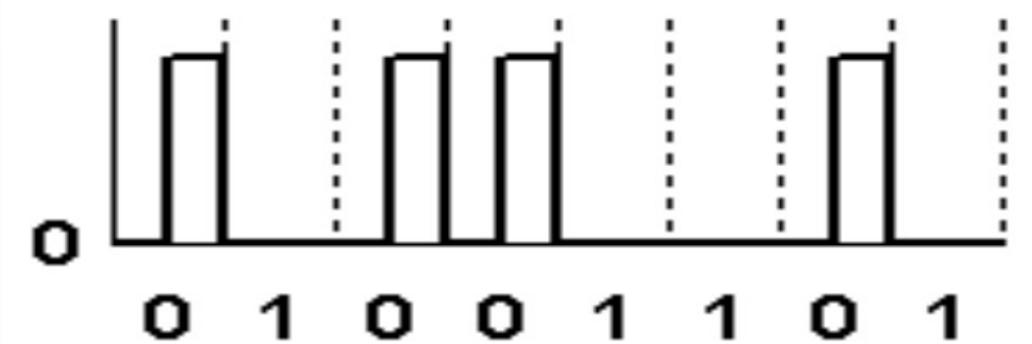
- Modulační rychlosť / Nyquistovo kriterium
  - $v_m = 2 * W [ Bd ]$
- Maximální přenosová rychlosť pro kanál bez šumu / kapacita přenosového kanálu
  - Vychází z Nyquistova kritéria
  - $v_p = 2 * W * \log_2(V) [ b/s ] , v_p = v_m * \log_2(V) [ b/s ]$
- Maximální přenosová rychlosť pro kanál s šumem / kapacita přenosového kanálu
  - $v_p = W * \log_2(1+S/N) [ b/s ]$
- Využití kapacity přenosového kanálu – například pro arytmický přenos
  - $n = N/M * 100 [ \% ]$

# L1: Přenos v základním pásmu

- Přenos v základním pásmu / nemodulovaný přenos
  - Měníme signál přímo dle dat a odečítáme jeho konkrétní hodnotu
  - Například úroveň napětí
  - Nejedná se o ideální přenos, protože „obdélníkový“ signál se nepřenáší ideální
    - Problém jsou především ostré hrany, které přenos „zaobluje“
  - Používá se při přenosu na kratší vzdálenosti, kde není zkreslení tak výrazné
  - Například v LAN( Ethernet )

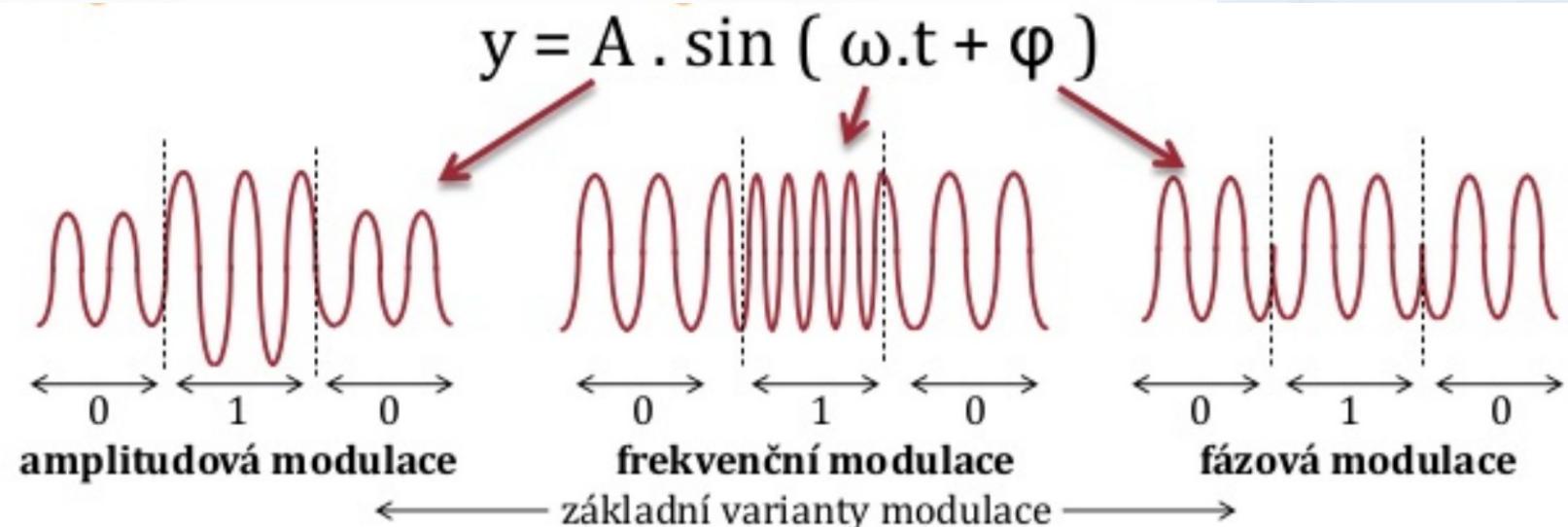


zdroj: <https://www.youtube.com/watch?v=i8CmibhvZ0c>



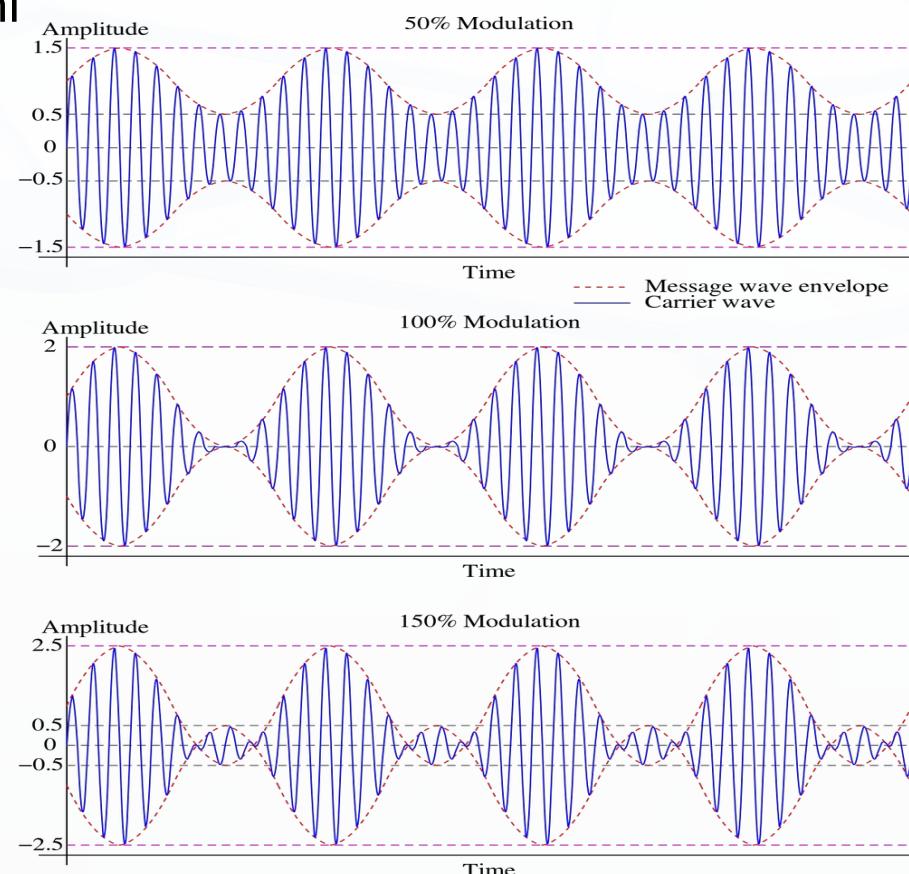
# L1: Přenos v přeloženém pásmu

- Přenos v přeloženém pásmu / Modulovaný přenos
- K přenosu se využívá harmonický signál, který se kanálem šíří lépe
  - Jedná se o nosný signál – sám nenesí data, ta jsou „přidána“ až prostřednictvím změn signálu: **Modulací**
- Přenášený signál „měníme/moduluje“ tak, aby byl lépe zřetelný rozdíl mezi jednotlivými stavami
- Přenášený signál je složením dle:  $y = A \cdot \sin ( w.t + f )$ 
  - A – amplitudová složka
  - w.t – frekvenční složka
  - f – fázová složka
- Modulace tedy může být:
  - Amplitudová
  - Frekvenční
  - Fázová
  - Kombinace všech nebo části předchozích jmenovaných



# L1: Modulace signálu: Amplitudová

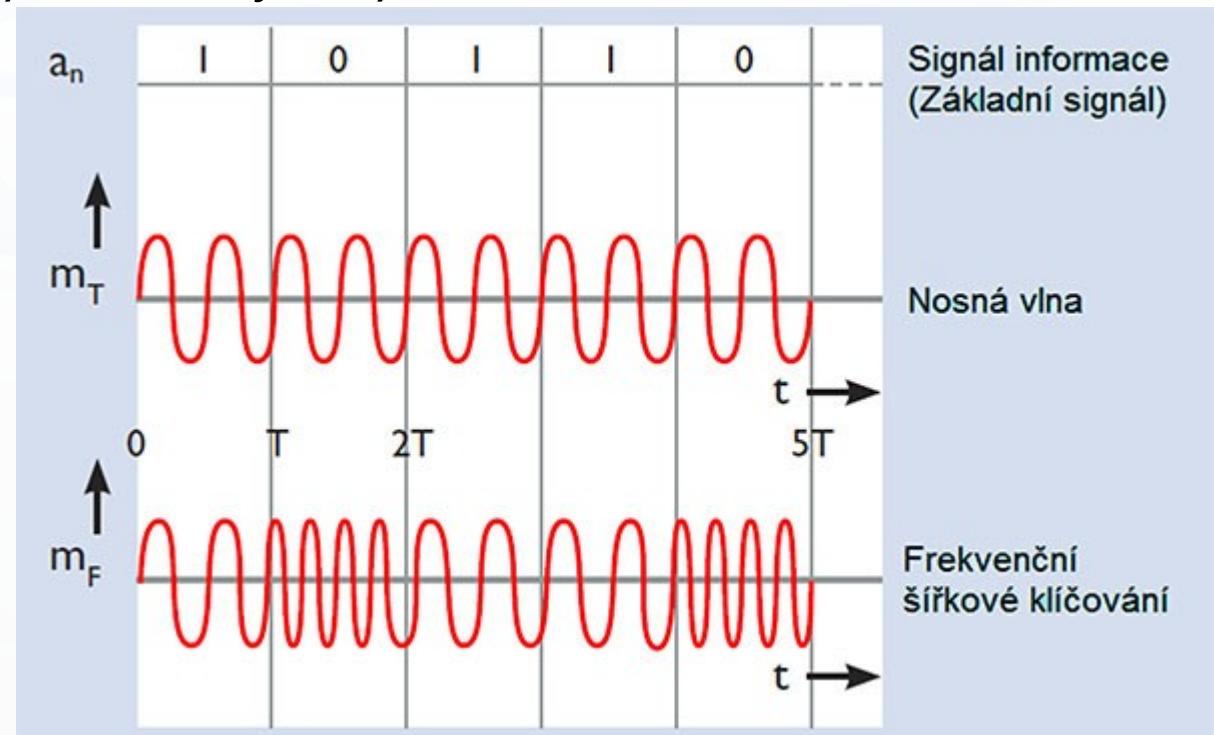
- AM - Měníme amplitudu signálu k rozlišení stavů
- Ostatní parametry, tedy frekvence a fáze se nemění
- Rozlišujeme dva nebo více stavů dle úrovně amplitudy



zdroj: <https://commons.wikimedia.org>

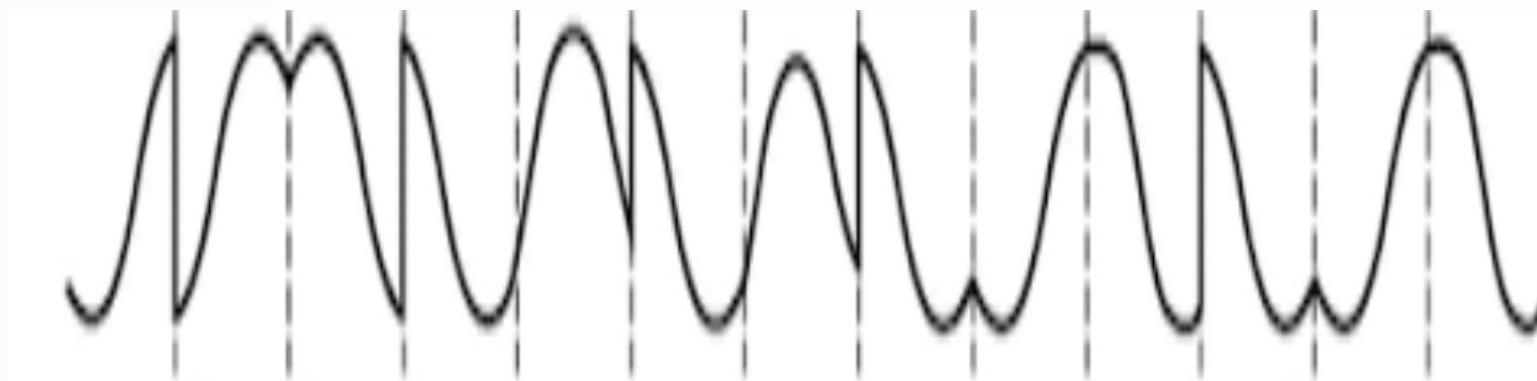
# L1: Modulace signálu: Frekvenční

- FM – Mění se frekvence signálu
- Ostatní parametry, tedy amplituda a fáze se nemění
- V základu opět rozlišujeme dva stavy, ale může jich být i více



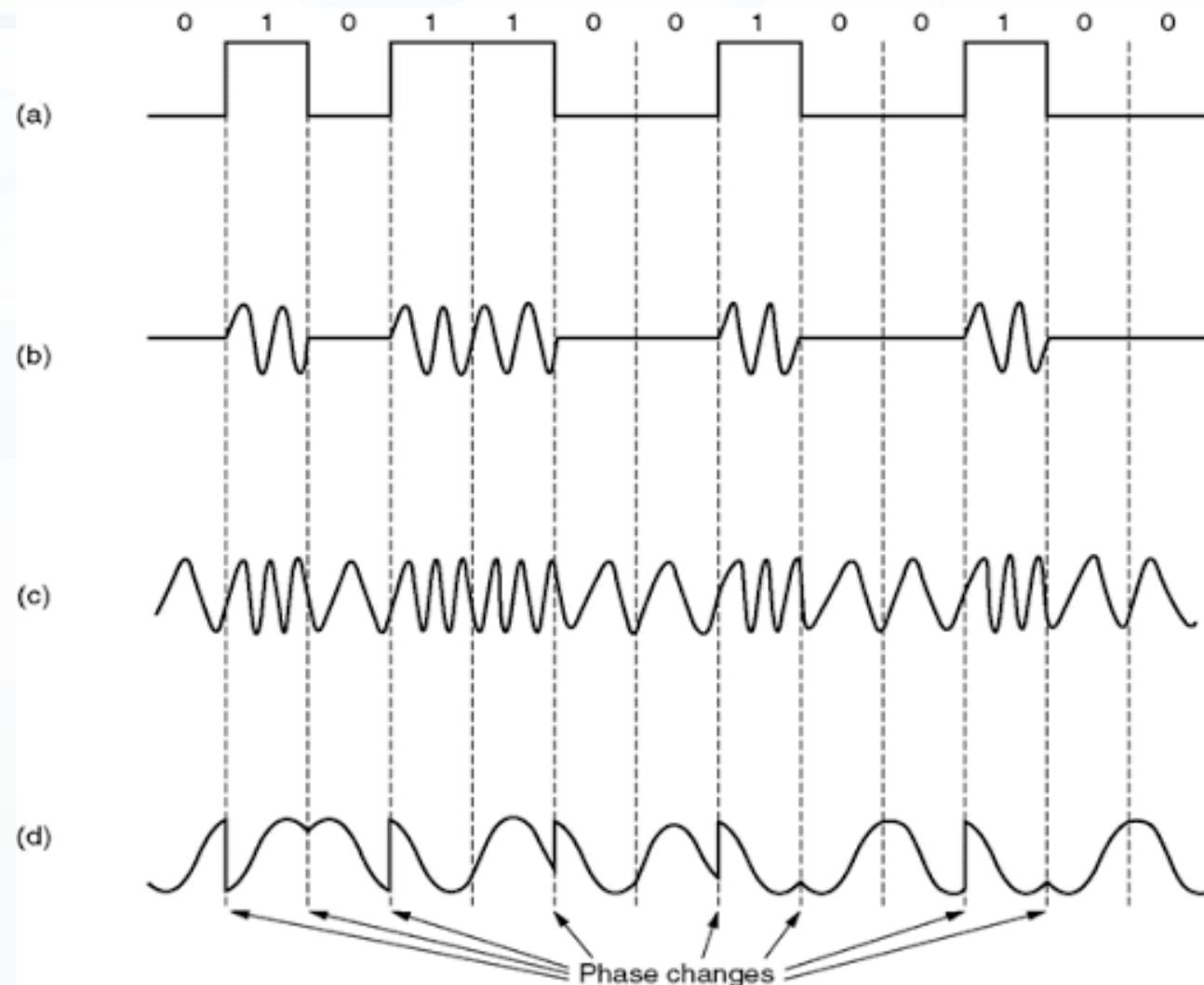
# L1: Modulace signálu: Fázová

- PM – Fázová modulace
- Ostatní parametry, tedy amplituda a frekvence se nemění
- Změnu hodnoty signálu ( 0,1 ) signalizuje změna fáze
  - Pokud se fáze nezmění, nemění se ani přenášena hodnota
  - Jinak řečeno bez posunu fáze při taktu hodin přenášíme stejná data



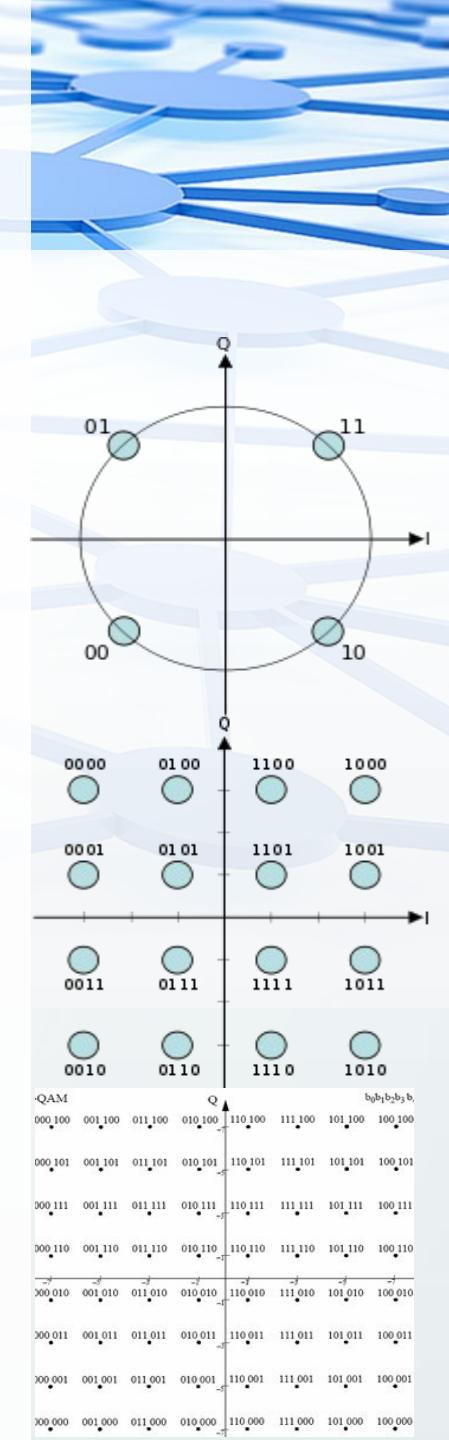
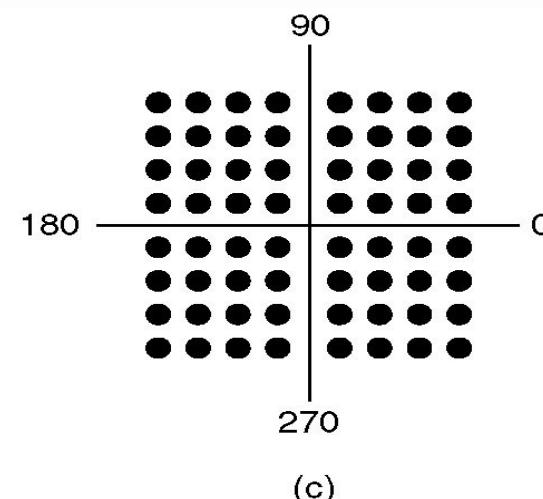
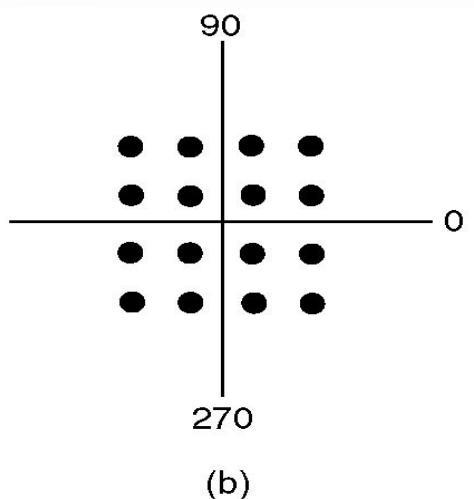
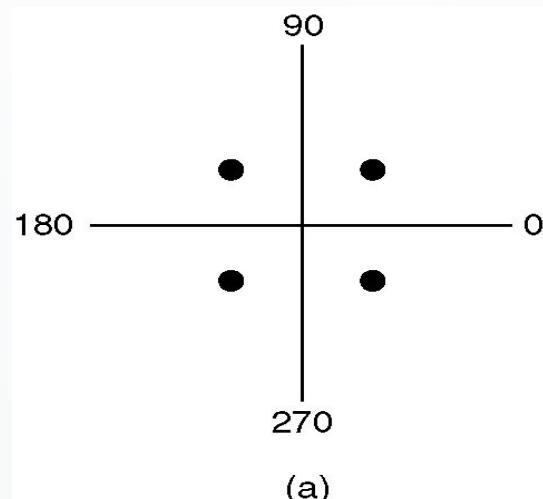
zdroj: <http://www.cs.vsb.cz/grygarek/PS/lect/fyzPrincipy.html>

# L1: Modulace signálu: Porovnání



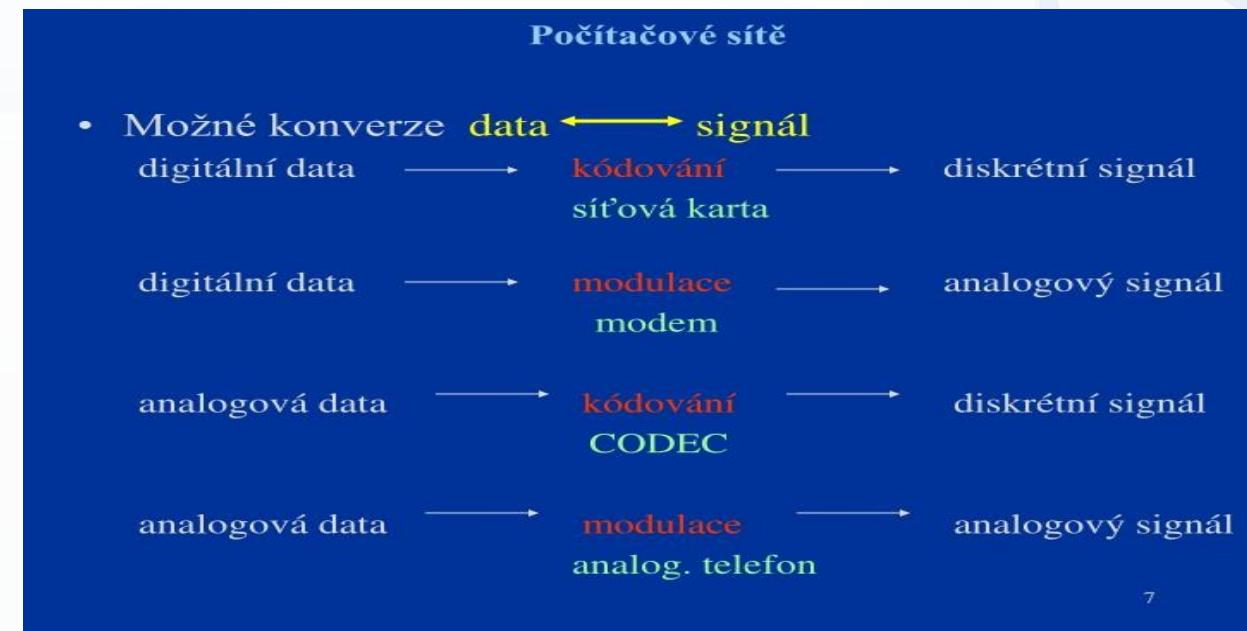
# L1: Modulace signálu: Kombinované modulace

- V praxi se často modulace kombinují, tedy nemění se jen jedna složka, ale více složek z: amplituda, frekvence, fáze
  - QPSK ( a ) - schodný konstelační diagram s QAM-4, umožňuje kódovat 2 bity na symbol
  - QAM-16 ( b ) - využívá dva nosné signály se stejnou frekvencí, posunuté o 90 stupňů
    - Jeden signál modulován amplitudově, druhý signál modulovaný frekvenčně
    - 4bity na jednu změnu
  - QAM-64 ( c ) - 6 bitů na jednu změnu



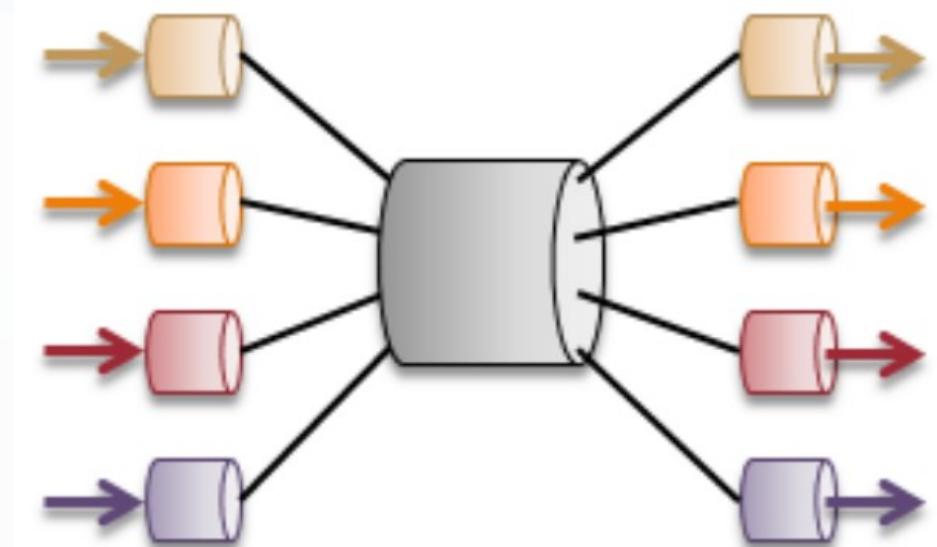
# L1: Konverze signálů

- Na různých místech v síti dochází k různým přenosům
  - V základním či přeloženém pásmu
- Tyto přenosy umožňují různá zařízení / software
  - Síťová karta - přenos v základním pásmu
  - Modem - přenos v přeloženém pásmu
  - Kodek a „analogový telefon“
    - Opačný problém jak přenést analogová data



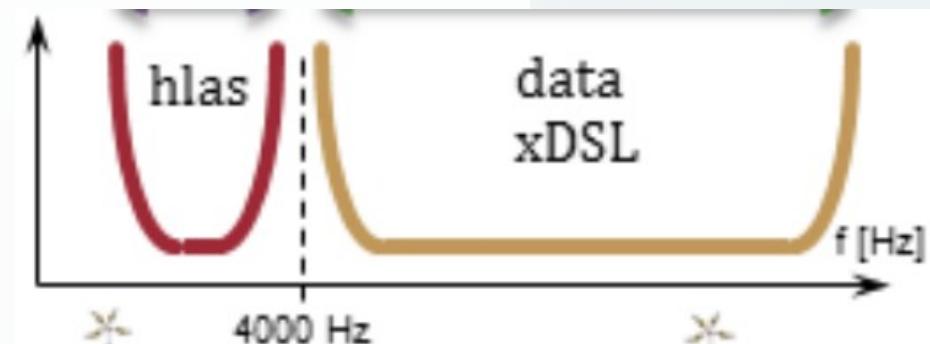
# L1: Multiplex

- Multiplex - způsob jak realizovat více „paralelních“ přenosů jednou datovou cestou – jedním přenosovým kanálem
- Dva důvody použití
  - Mám pouze jedinou přenosovou cestu, ale potřebuji přenést více samostatných signálů
    - Data cesta nemusí být trvale kompletně využita a tedy se vyplatí ji sdílet
  - Mám přenosovou cestu, jejích parametry umožňují realizovat více přenosů
    - Šetřím zdroje bez omezení rychlost
- Základní dělení multiplexů
  - Analogový – vychází z fyzikálních možností přenosového kanálu
    - Frekvenční
    - Vlnový
  - Digitální – nejedná se o „plné“ sdílení, ale dodávám další logiku, která sdílení umožňuje
    - Časový
    - Statistický
    - Kódový



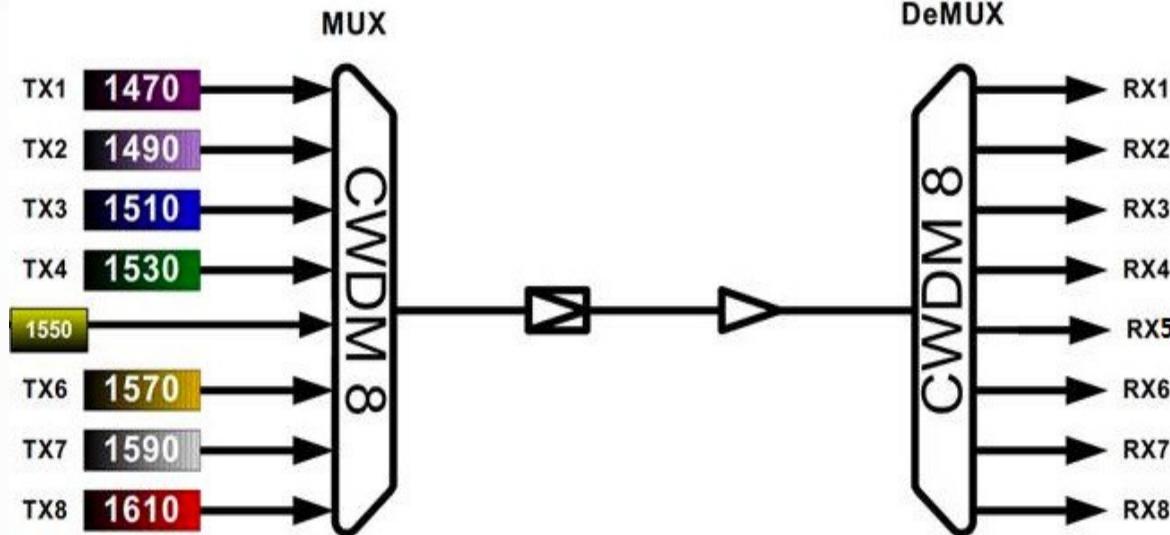
# L1: Multiplex: Frekvenční

- FDM - Frekvenční multiplex je typ analogového multiplexu
  - Přenášíme různé signály s různou frekvencí
- Vycházíme ze dvou předpokladů
  - Přenášený signál je harmonický a frekvenčně omezený – potřebuje jen určitou „úzkou“ šířku pásma
  - Přenosová cesta disponuje větší šírkou pásma a tedy může přenášet více signálů různých frekvencí
    - Může virtuální přenosový kanál rozdělit na sloty dle frekvencí a jednotlivým přenosům vyhradit konkrétní sloty
  - Definice šířky pásma kanálu i velikosti slotů jsou konstantní ( ale nemusí být stejně velké )
  - Jednotlivý sloty na sebe nemohou „pevně doléhat“
    - Je nutný rozestup kvůli přeslechům – nedokonalostem použité technologie
- Použití v praxi:
  - Analogové rozhlasové a televizní vysílání
  - V telefonní síti k oddělení hlasových a datových přenosů



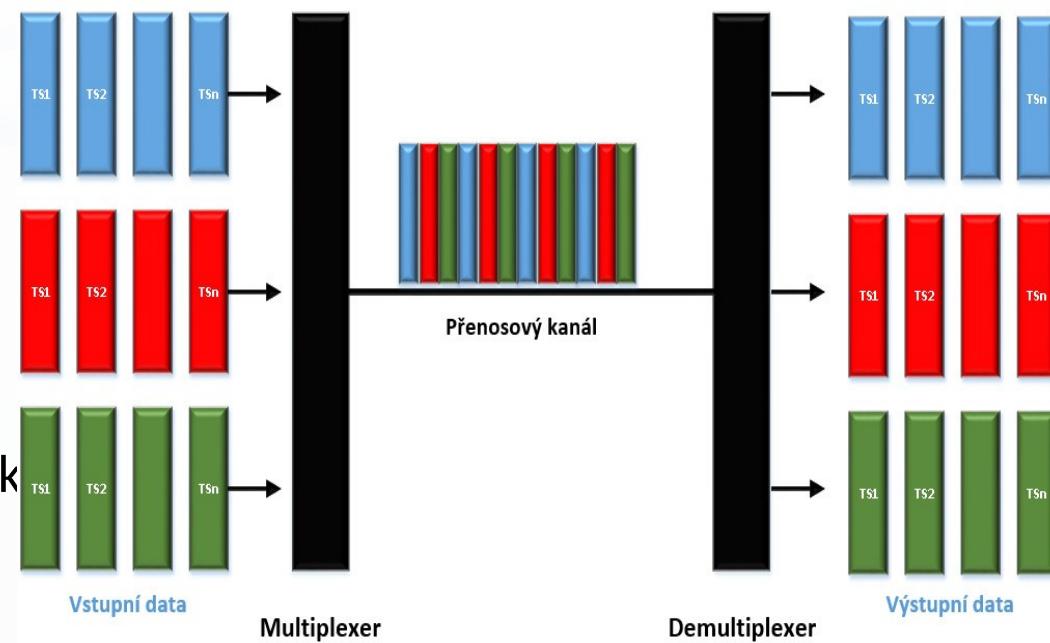
# L1: Multiplex: Vlnový

- WDM – Vlnový multiplex je typ analogového multiplexu
  - Přenášíme různé signály s různou vlnovou délkou - „jinak barevné signály“
- Vycházíme z předpokladu, že přenosový kanál – např optické vlákno – dokáže přenášet světelný signál o různých vlnových dálkách a že jsme schopni tyto signály od sebe odlišit
- Velmi časté použití v páteřních síťových technologiích
  - Můžeme navýšovat kapacitu přenosového kanálů bez nutnosti pokládky dalších vodičů



# L1: Multiplex: Časový

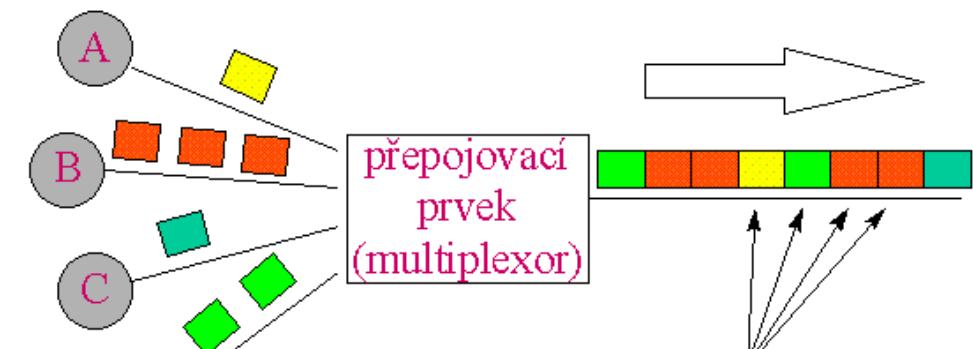
- TDM – Časový multiplex je typ digitálního multiplexu
  - Reálně neumožnuje paralelní přenosy, ale sdílí přenosové medium
- Vycházíme z předpokladů:
  - přenosový kanál je schopen přenášet jen jeden signál
  - není možné zajistit další přenosové kanály
  - V ideálním případě není přenosový kanál trvale využity na 100%
- Realizace:
  - přenosový kanál je rozdělen na pevně daná časová pásma
    - kvanta času
  - Každý účastník dostane přidělené konkrétní časové pásmu, kdy může vysílat
- Vzniká problém při různé intenzitě vysílání jednotlivých účastníků
  - Někdo chce vysílat, ale musí čekat až na něj přijde řada
  - Jiný účastník vysílá řídce a jeho časový slot je nevyužity



zdroj: <https://docplayer.cz/38882541-Metody-prenosu-a-spojovani-pro-integrovanou-vyuku-vut-a-vsbtuo.html>

# L1: Multiplex: Statistický

- STDM – Statistický multiplex je typ digitálního multiplexu
  - Řeší problém časového multiplexu pro kolísavou hustotu vysílání
- Vycházíme z předpokladů:
  - přenosový kanál je schopen přenášet jen jeden signál
  - není možné zajistit další přenosové kanály
  - V ideálním případě není přenosový kanál trvale využity na 100%
  - Různí účastníci vysílají různě často – s různou statistikou
- Realizace:
  - přenosový kanál je rozdělen na pevně daná časová pásma
    - Stejně jako u časového multiplexu
  - Žádný účastník nemá pevně přidělený čas, ale může vysílat kdykoliv na něj vyjde řada
  - Je nutné jednotlivé zdroje vhodně oddělit
    - Například označení kódem na začátku dat
- Řeší problém kolísavé potřeby vysílání
- Má lepší využití přenosové kapacity
- Negarantuje přístup k přenosu do určitého času



jednotlivé kanály nemají pevně přiřazené časové sloty, jejich data proto musí být vhodně identifikována

# L1: Multiplex: Kódový

- CDM – Kódový multiplex je typ digitálního multiplexu
- Vycházíme z předpokladů:
  - Je vhodné využít v maximální možné míře přenosový kanál
  - Je možné přenášet data od všech účastníků naráz a příjemce si vybere
- Realizace:
  - Data data všech účastníků jsou přenášena naráz všem příjemcům
  - Data jsou „vhodně“ kódována a příjemce si vybere jen ta co potřebuje
  - Data musí být vhodně kódována, aby bylo možné jednotlivé zprávy dekódovat
  - Kódovaná data musí být vzájemně ortogonální
    - Mají nulový skalární součin => jsou na sobě zcela nezávislé
- Používá se mobilních sítích - technologie CDMA



zdroj:<https://www.eearchiv.cz/a96/>

# L1: Zpoždění v přenosech

- Zpoždění signálu je nežádoucí, ale logický jev v přenosu dat
  - Bylo by ideální kdyby byla data přenesena „okamžitě“, ale to je nereálné
  - Přenos je omezen fyzikálními vlastnostmi signálu a vodiče, kódováním, modulací, zpracováním na straně příjemce, zahlceností datové cesty atd.
- Zpoždění můžeme dělit na:
  - Zpoždění při zpracování – nemění se, lze predikovat
    - Zpoždění signálu
      - Jak dlouho trvá než se signál dostane z jednoho konce přenosového media na druhý
      - Závisí na rychlosti šíření signálu – například světla
      - Závisí na délce přenosového media
    - Zpoždění přenosu
      - Jak dlouho trvá přenos celého bloku dat
      - Závisí na přenosové rychlosti – b/s
      - Nezávisí na délce přenosového media
  - Zpoždění ve frontách – mění se v čase, není možné predikovat
    - Vstupní fronta přijímače
    - Fronta na zpracování na CPU
    - Výstupní fronta vysílače

# L1: Latence a RTT

- Obojí vypovídá o mře zpoždění – a tedy o kvalitě přenosové cesty
  - Obojí chceme mít co nejnižší
- Latence – míra zpoždění ( jednosměrná )
  - Jak dlouho trvá jedna cesta data včetně zpracování od vysílače k přijímači
  - Různá přenosová media a technologie mají různou
    - GSM: až 800ms, LTE: pod 100ms, xDSL: desítky ms, Ethernet: kolem 0,3 ms
  - Různé technologie potřebují nejhůře nějakou míru latence aby správně fungovaly
    - Telefonie: do 200ms, on-line hry: méně než 100ms, ....
- RTT – Round Trip Time - „doba obrátky“
  - „obou směrná latence“
    - Jako dlouho trvá datům cesta tam a zpět
    - Je závislá na velikosti dat – logicky přenos více dat trvá déle
    - Může jej testovat pomocí ICMP – příkazem ping

```
Microsoft<R> Windows DOS
<C>Copyright Microsoft Corp 1990-2001.

C:\DOCUME^1\CHAD\DESKTOP>ping www.youtube.com

Pinging youtube-ui.l.google.com [74.125.127.113] with 32 bytes of data:
Reply from 74.125.127.113: bytes=32 time=53ms TTL=247
Reply from 74.125.127.113: bytes=32 time=55ms TTL=247
Reply from 74.125.127.113: bytes=32 time=54ms TTL=247
Reply from 74.125.127.113: bytes=32 time=53ms TTL=247

Ping statistics for 74.125.127.113:
  Packets: Sent = 4, Received = 4, Lost = 0 <0% loss>,
  Approximate round trip times in milli-seconds:
    Minimum = 53ms, Maximum = 55ms, Average = 53ms

C:\DOCUME^1\CHAD\DESKTOP>
```

# L1:Jitter

- Jitter – rozptyl nebo také kolísání parametrů přenosu
  - Defakto určuje míru nestability přenosu z pohledu parametrů přenosu
- Možno měřit tu latence i RRT
- Často je vyjadřován jako dvojice hodnot - min a max nad měřenými daty
- Opět, stejně jako u latence a RTT chceme co „nejnižší“, ale z jiného důvodu
  - Nízká latence či RTT vypovídají o kvalitě přenosu z pohledu rychlosti
  - Jitter neříká o rychlosti nic, ale říká nakolik se dá na současný stav parametrů sítě spolehnout
- Nízký rozptyl je zásadní například u multimediálních či hlasových přenosů
  - Potřebujeme aby data přicházela pravidelně a trvale, aby nedocházelo k „trhání“ hlasu či obrazu

# L1: Přenosová media

- Vedená / drátová media, vedena hmotným materiélem, snadnější zacílení signálu
  - Kroucená dvojlinka
  - Koaxiální kabel
  - Optické vlákno
- Nevedená / bezdrátová, volně se šířící prostorem či hmotou ( vzduch, voda, vakuum )
  - Rádiové
  - Infračervené
  - Optické

# L1: Přenosová media – kroucená dvojlinka



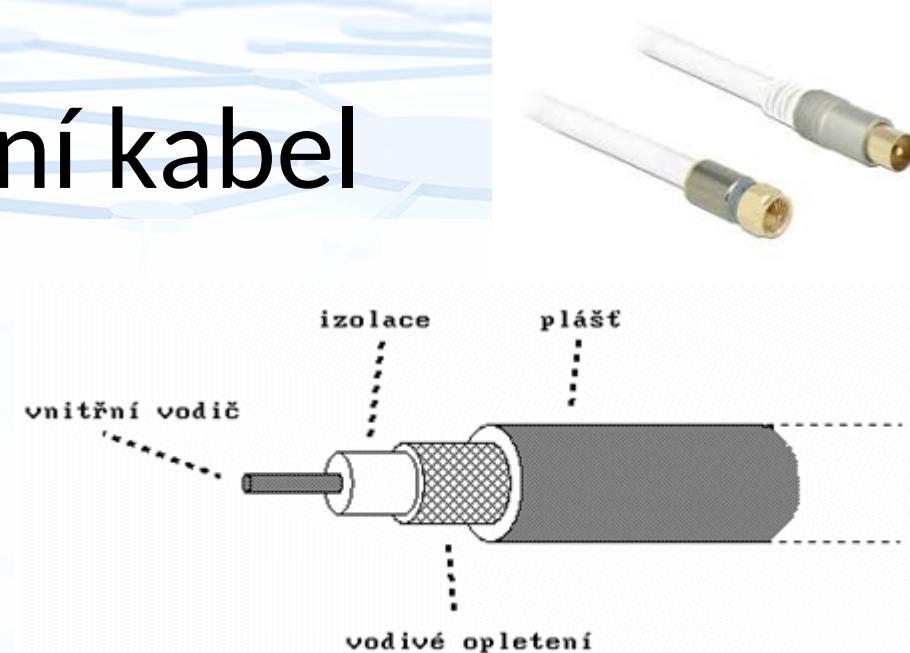
- Základní princip a parametry představeny na první přednášce
- Mechanicky velice odolná
- Čím lepší stínění, provedení a materiál, tím lepší parametry a tím i dostupná přenosová rychlosť
- Typické využití v LAN
- Dnes nejběžněji používaný vodič pro počítačovou síť

Category	Standard	Data rate	Frequency	# of Conductors
Cat 5	100BASE-TX	100 Mbit	100 MHz	4 or 8
Cat 5e	1000BASE-TX	1 Gbit	100 MHz Duplex	8
Cat 6	EIA/TIA 568B2.1	1-10 Gbit*	250 MHz	8
Cat 6A	10GBASE-T	10 Gbit	500 MHz	8
Cat 7	10GBASE-T	10 Gbit	600 MHz	8
Cat 7A	10GBASE-T	10 Gbit	1000 MHz	8
Cat 8	40GBASE-T	40 Gbit	1600-2000 MHz	8

\* Depends on length and cable type

# L1: Přenosová media – koaxiální kabel

- Základní princip a parametry představeny na první přednášce
- Mechanicky velice odolná
- Obecně lepší parametry pro přenos než kroucená dvojlinka
  - Především kvůli vodivému opletení, které zajišťuje lepší stínění
- Dříve používaný samostatně pro LAN a Ethernet ( 10Base5, 10Base2 ) a dálkových kabelech – např podmořské vedení
- Dnes typicky použitý pro kabelové televize, jejichž rozvody se používají jako součást počítačové sítě mezi ISP a LAN

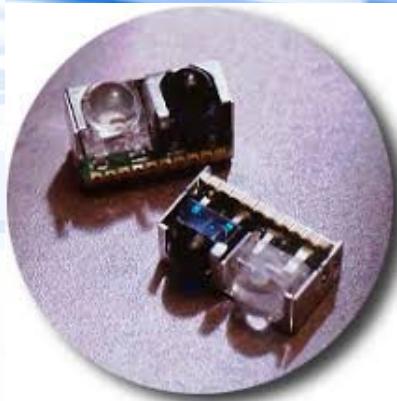


zdroj:<https://www.eearchiv.cz/b05>

Cable picture					
Description	Sucoform_86 Koaxialkabel, Low Loss	RG196_A/U Koaxialkabel	LN 5001 Koaxialkabel, Low Noise Kabel	RGL196 Koaxialkabel, Low Noise	
Electrical values					
Impedance	[Ω]	50 Ω	50 Ω	50 Ω	50 Ω
Max. frequency	[Ghz]	40	1	3	2.5

zdroj:<https://www.koax24.de/en/product-info/coaxial-cable/coaxial-cable-50-ohm/15-21-mm-size1/rг405-u.html>

# L1: Přenosová media - infračevené spoje - IrDA



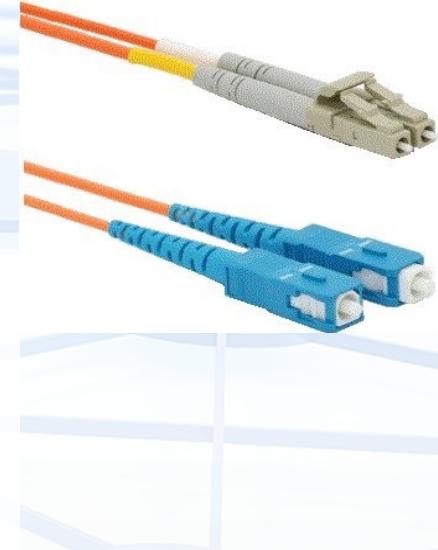
- Nemá žádné vodící medium, ale šíří se prostorem
- Jedná se o infračervené signály
  - Používá se LED diod v infra spektru (880nm)
- Vyžaduje přímou viditelnost – směrování zařízené na sebe
- Typicky velice malý dosah – max desítky centimetrů
- Často používán na dálkových ovladačích
- V současné době počítačových přenosech nahrazován Bluetooth
  - Má delší dosah – až metry a nevyžaduje přímou viditelnost
- Maximální rychlosť 4Mbps při frekvenci cca 1.15 MHz

# L1: Přenosová media – optické spoje

- Bezdrátový optický spoj - FSO, Free Space Optics
- Nemá žádné vodící medium, ale šíří se prostorem
- Jedná se typicky o laserová pojítka
- Vyžaduje přímou viditelnost – směrování zařízené na sebe
- Vyžaduje velice přesné zaměření
- Velice odolné vůči odposlechům či rušení
  - Paprsek je velice tenký a není vidět
  - Využívá frekvence 1GHz a více
- Typické propojení jednotlivých budov
- Maximální přenosové rychlosti až do 10Gbps
- Maximální přenosová vzdálenost až 15km



# L1: Přenosová media – optická vlákna



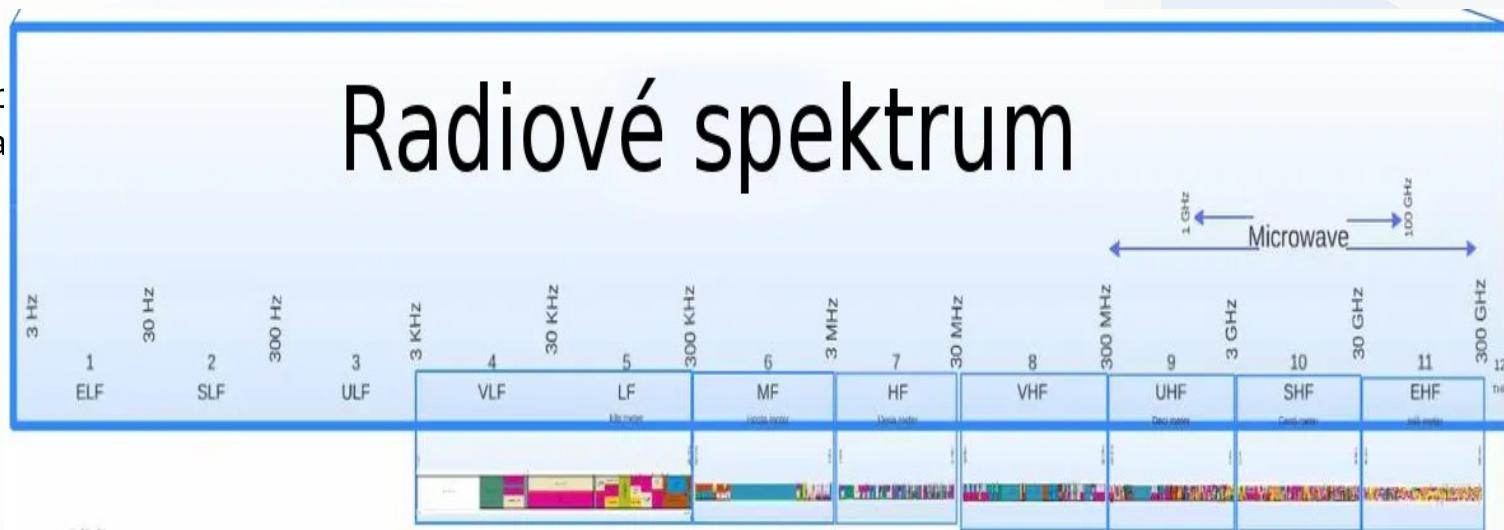
- Základní princip a parametry představeny na první přednášce
- Mechanicky velice křehké – platí pro křemíková vlákna
  - Na malé vzdáleností je možné křemík nahradit plastem
    - Horší parametry, ale vyšší odolnost a nižší cena
- Nulová interference s okolím
- K přenosu se typicky využívá infračervené světlo v pásmech 850 - 1550nm
  - Má nižší útlum než viditelné světlo
- Dvě základní provedení
  - Jednovidová – 1310 - 1550 nm
    - Jeden přesný signál na dlouhé vzdálenosti, bez odrazu
    - Nasvěcováná laserem
    - 100-1000Gbps
  - Multividová 850 – 1310 nm
    - Více signálu v jednom vlákně, využívá odrazu
    - Vlnový multiplex
    - K nasvěcování stačí LED
    - Do 100Gbps – na kratší vzdáleností ~ 100m
- Převod
  - Vlnová délka ( $\lambda$ )  
$$\lambda = 300 / f [m]$$
  - Frekvence/kmitočet  
$$f = 300 / \lambda [\text{MHz}]$$



# L1: Přenosová media - radiové spoje



- Nemá žádné vodící medium, ale šíří se prostorem
- Jedná se o radiové nebo mikrovlnné signály
- Signál který přenášíme je typicky úzký – omezen šírkou pásma
- Dva možné způsoby vysílání
  - Přenos úzkopásmový
    - Využívá jen takovou šířku pásma, kterou skutečně potřebuje
    - Citlivý na rušení, snadný odposlech
  - Přenos v rozprostřeném spektru
    - Využívá větší šířku pásma než skutečně potřebuje
    - Mění frekvence na kterých je přenos realizován
    - Odolnější vůči rušení i odposlechu
    - Více technik rozprostření
      - FSSS, DSSS, FDM, UWB
- Typické využití od místních WLAN po středně dlouhé spoje
  - Max v malých jednotkách kilometrů
  - Nutné přímá viditelnost

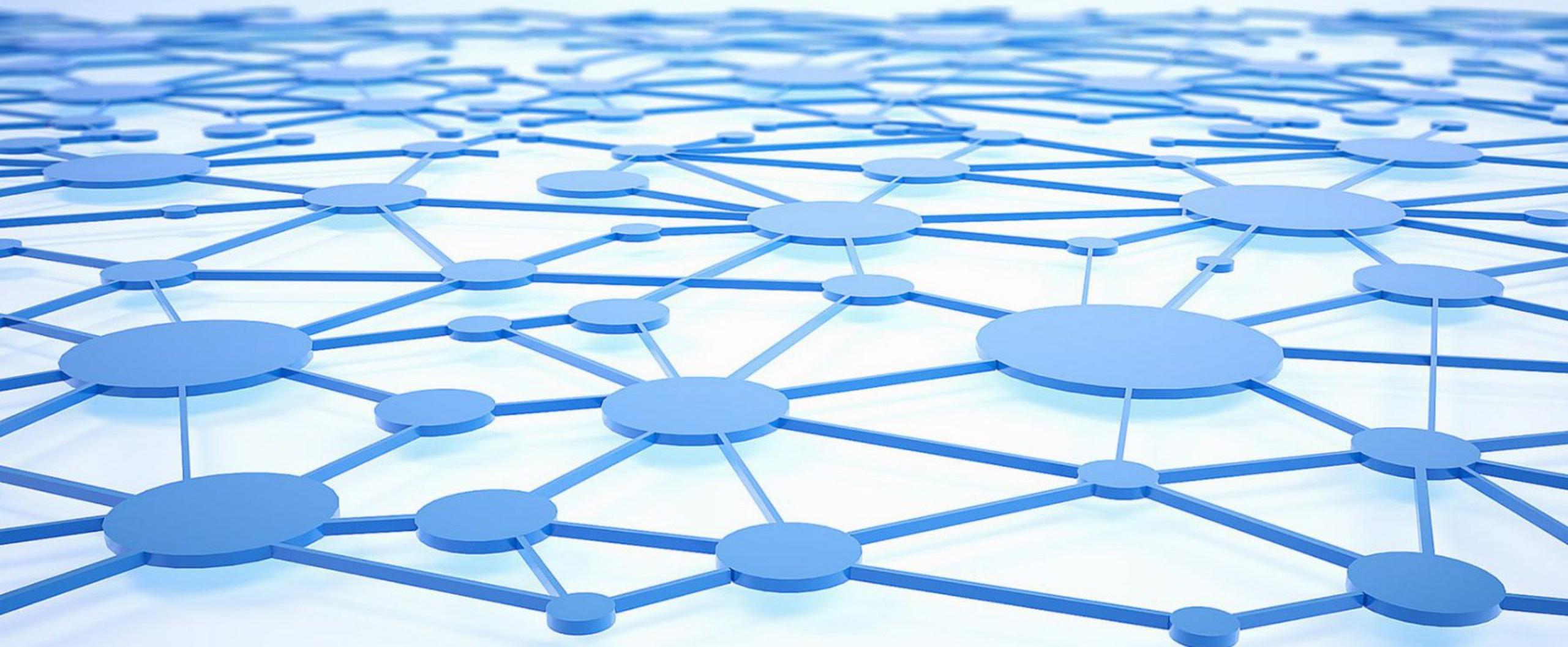


zdroj:<http://www.fader.cz/2017/11/06/bezdratove-mikrofony-vs-televizni-vysilani-tabulka/>

# Úvod do počítačových sítí

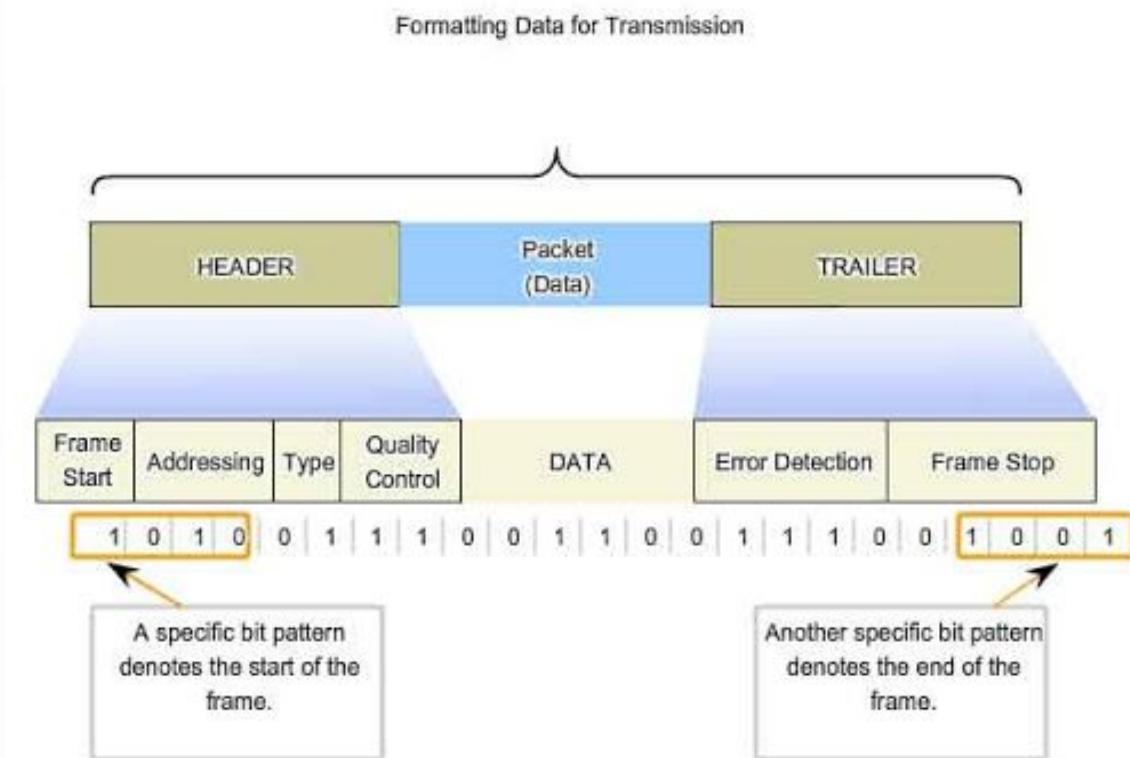
Přednáška 5 ( 2025/2026 )

ver. 2025-10-15-01



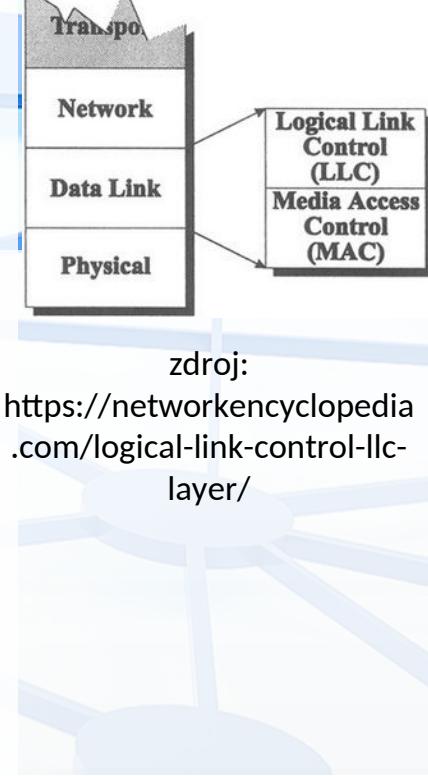
# Linková vrstva: L2

- Základní funkce L2 – linkové vrstvy
    - Přijímá data od L1 vrstvy / předává data L1 vrstvě ( bity )
      - Využívá služeb L1 k přenosu rámců
    - Provádí rozdělení na rámce
    - Provádí kontrolu správnosti přenosu
    - Detekuje a řeší chyby v přenosu
    - Řídí přístup k komunikačnímu kanálu
    - Řídí rychlosť přenosu
    - Zajišťuje přenos rámců v rámci LAN / mezi sousedními uzly
    - Předává data L3 vrstvě / přijímá data od L3 vrstvy ( pakety )
      - Poskytuje služby L3 – přenos data, řešení chyb, řízení rychlosti



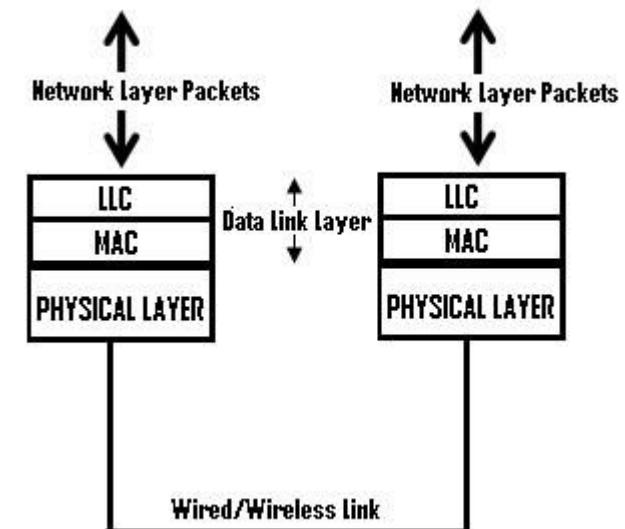
# L2: Dělení linkové vrstvy

- L2 v ISO/OSI je „přetížená“
  - Má více rozdílných funkcí
- Začala se pro přehlednost dělit na dvě části – dle funkce
  - MAC - Media Access Control
    - fyzické adresování,
    - řízení přístupu k médiu
    - Fragmentace/defragmentace data ( bit → rámec )
  - LLC - Logical Link Control
    - řízení toku
    - zabezpečení proti chybám
    - zabalování / rozbalování L3 paketů



zdroj:

<https://networkencyclopedia.com/logical-link-control-llc-layer/>



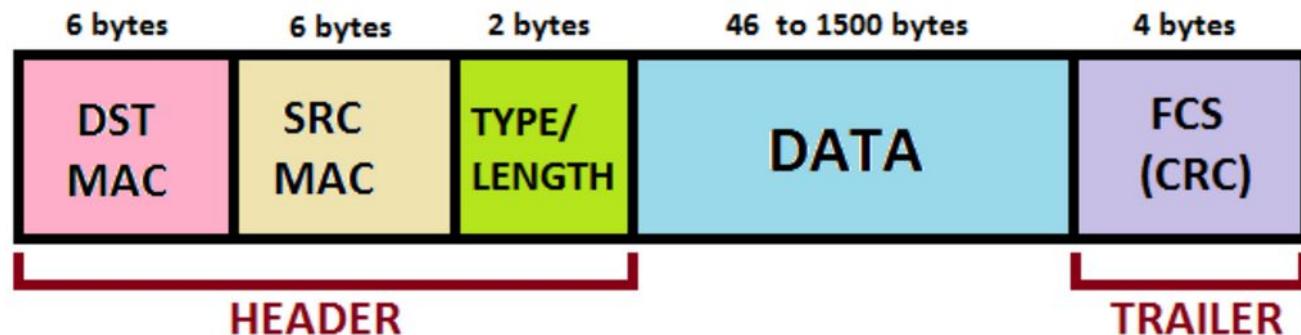
zdroj: <http://computernetworkingsimplified.in/data-link-layer/components-data-link-layer-llc-mac/>

# L2: Rámce a jejich význam

- Rámcem je základní datová jednotka L2 vrstvy
- Rámce mohou být
  - Datové – obsahují data, které chceme přenést
  - Řídící – obsahují příkazy, potvrzení či další informace nutné k řízení přenosu
- Framing – rozdělení streamu 0 a 1 od L1 na rámce – frame
- L2 už „nepřenáší“ jednotlivé bity, ale celé rámce
  - Která se z 0 a 1 skládají
- Konkrétní rámcem je vázaný na konkrétní L2 protokol – je vždy jiný
  - Ethernet, Token Ring, PPP, SLIP, ....

# L2: Rámce a jeho struktura

- Rámcem se liší dle konkrétního L2 protokolu, ale základní struktura je stejná
- Hlavička
  - Cílová L2 adresa
  - Zdrojová L2 adresa
  - Identifikace protokolu / délka rámce /... liší se dle protokolu
- Data
  - Datový rámcem - L3 paket
  - Řídící rámcem - L2 data pro řízení přenosu, např ACK potvrzení správnosti přenosu
- Patička - FCS - frame check sequence
  - Zabezpečení rámce - ověření správnosti přenosu
    - Parita, checksuma, CRC, ...



zdroj: <https://www.bitforestinfo.com/2018/01/code-ethernet-ii-raw-packet-in-python.html>

# L2: Detekce rámce

- Detekce rámce je jednou ze základních funkcí L2 vrstvy
  - Od L1 dostává nestrukturovaný stream/proud 0 a 1
- Metody detekce záleží na typu přenosu
  - Odpočítat rámcem
    - Pokud data jsou vždy těsně za sebou ( bez mezer ) a mají pevně danou strukturu
    - Například v rámci T nebo E kanálů
    - Podobný princip jako časový multiplex
  - Vyznačení začátku a konce rámce
    - Pomocí speciální sekvence bitů/znaků/bytů označíme začátek konec
      - Ale musí řešit transparentnost přenosu – tedy jak tyto speciální značky přenést v datech ....
  - Vyznačení začátku a dopočítání konce rámce
    - Stejně jako v předchozím případě vyznačím začátek rámce a jelikož znám délku rámce, tak už jen odpočtu příslušný počet bitů
    - Výhoda je, že šetřím kapacitu kanálu, protože nepotřebuji ukončovací značku rámce
    - Délka může být pevně daná nebo přenášená v datech
    - Na rozdíl od „Odpočítávání rámcem“ data nemusí jít bezprostředně za sebou

# L2: Typy přenosů

- Podle toho jak přenášená data ( 0 a 1 ) nahlížíme, můžeme na přelomu L1 a L2 definovat tři typy přenosu
  - Bitové orientovaný
    - Neskládám z bitů jednotlivé znaky, ale pracuji přímo s jednotlivými bity
  - Znakově orientovaný
    - Jako základní jednotku neberu 1 bit, ale pracuji vždy s N-ticí bitů => znakem
    - Znak může být různě dlouhý - dle protokolu ( typicky 7-8 bit, ale klidně např 5 )
  - Bytově orientovaný
    - Speciální případ znakově orientovaného přenosu, kdy délka jednoho znaku je 8 bitů - 1 byte

# L2: Bitový přenos

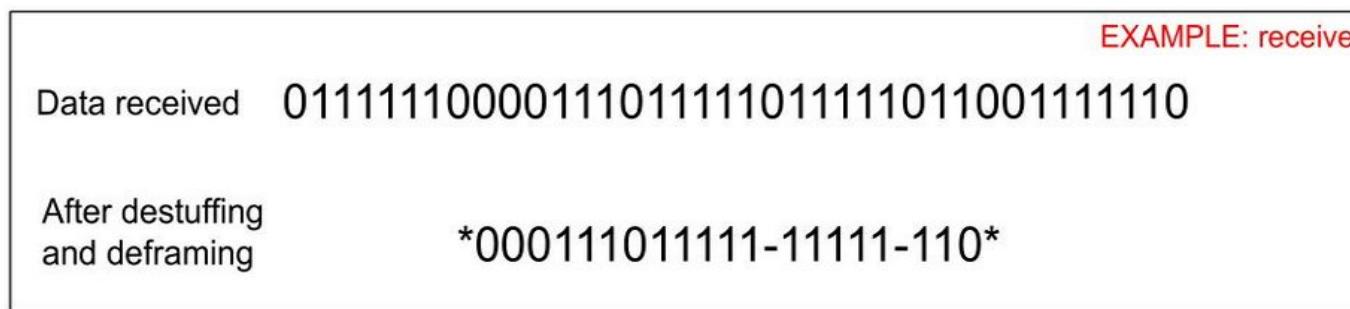
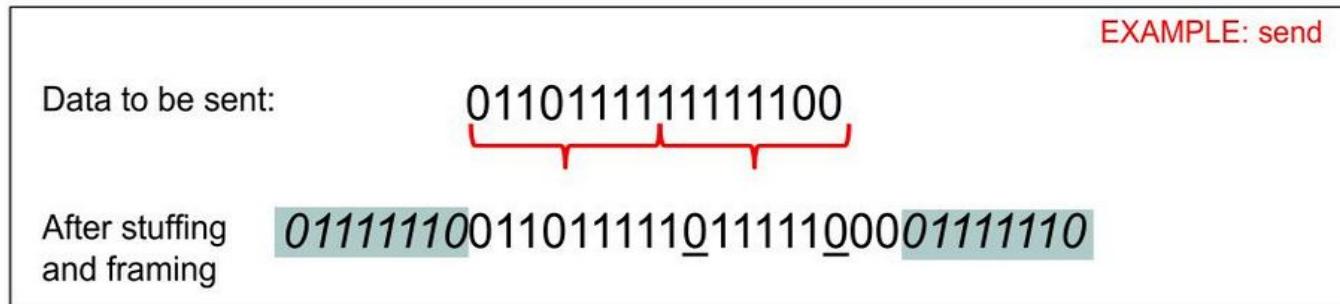
- Bitový orientovaný přenos bere jako základní jednotku 1 bit
- Tím že přenášíme jednotlivé bity, musí hranici tvořit nějaká bitová sekvence
  - Což vypadá stejně jako u znakově orientovaného protokolu, ALE
  - „křídlová značka“ – začátek rámce je sekvence bitů, které postupně dostaváme jeden po druhém
    - Často označovaná jako „flag“
  - Další data stále přenášíme po jednotlivých bitech bez ohledu na délku křídlové značky
- Příklad křídlové značky pro protokol HLDC
  - 0111110
  - Tedy nula - šest jedniček - nula => začátek rámce

# L2: Bitový přenos – transparentnost přenosu

- Stejně jako u znakově orientovaného přenosu vzniká problém transparentnosti přenosu
- Máme křídlovou značku/flag
  - 0111110
- Jak jej přenést v datech ?
  - Uvodit jej nějakým znakem nemůžeme – máme jen jednotlivé byty
  - Uvodit jej jediným bitem také ne, protože to může být jen 0 a 1
- Řešením je „vkládání bitů“
  - Pokud mám křídlovou značku 0111110 musím zajistit aby se mi nikde v datech nevyskytla stejná sekvence
  - Tedy aby tam nebylo 111111 – šest jedniček za sebou
  - Řešením je za každou „pátou jedničku“ uměle vložit nulu
    - POZOR – jen v datech
    - Nula je vložena automaticky na straně odesilatele ( pokud se v datech vyskytne pět jedniček )
    - Nula po sekvenci pěti jedniček je na straně příjemce odebrána ( pořad se bavíme jen v datech )

# L2: Bitový přenos – transparentnost přenosu -příklad

## Bit stuffing example



===== Giuseppe Bianchi =====

Zdroj: <https://www.slideserve.com/virote/layer-2-framing-hdrc-high-level-data-link-control>

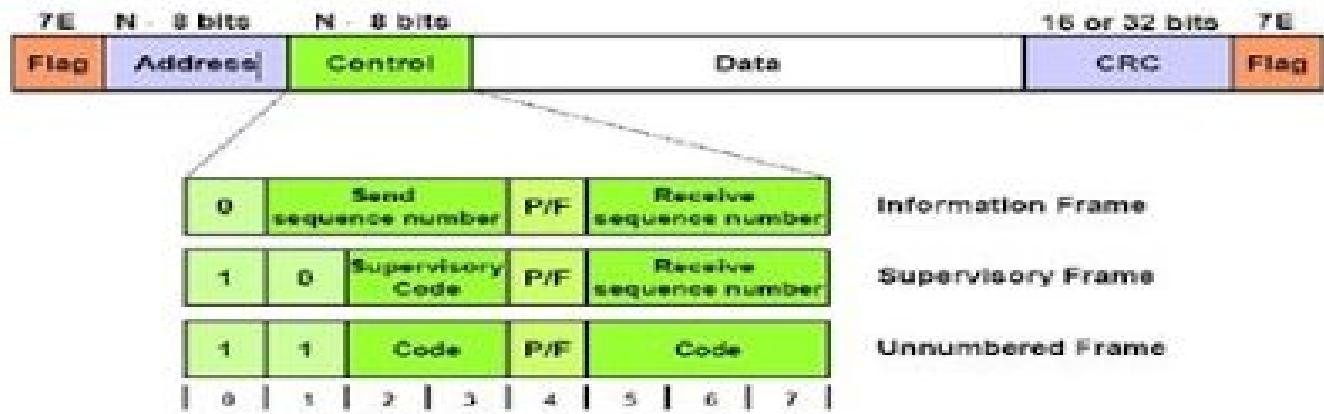
# L2: Bitově orientovaný přenos příklad - HLDC

- HLDC - High-Level Data Link Control
- Bitově orientovaný protokol pro Point-to-Point i Point-to-MultiPoint spoje
  - Sice už se dnes nepoužívá, ale sloužil jako „podklad“ nebo inspirace pro mnoho dalších protokolů
- Protokol používá křídlovou značku **01111110**
  - Křídlová značka na konci rámce může být i začátkem dalšího rámce => šetříme data
- Používá tři „typy“ rámců“, které se rozlišují v rámci části rámce „control“
  - Informační - 0 ( uživatelská data )
  - Řídící - 10 ( řídící informace )
  - Nečíslované - 11 ( vše ostatní )
- Obsahuje zabezpečení – FCS – patička
  - Používá **16 nebo 32 CRC** ( budeme mít v dálce v přednáškách )

# L2: Bitově orientovaný přenos příklad - HLDC - rámce

## HIGH LEVEL DATA LINK CONTROL

### HDLC FRAME FORMAT



10

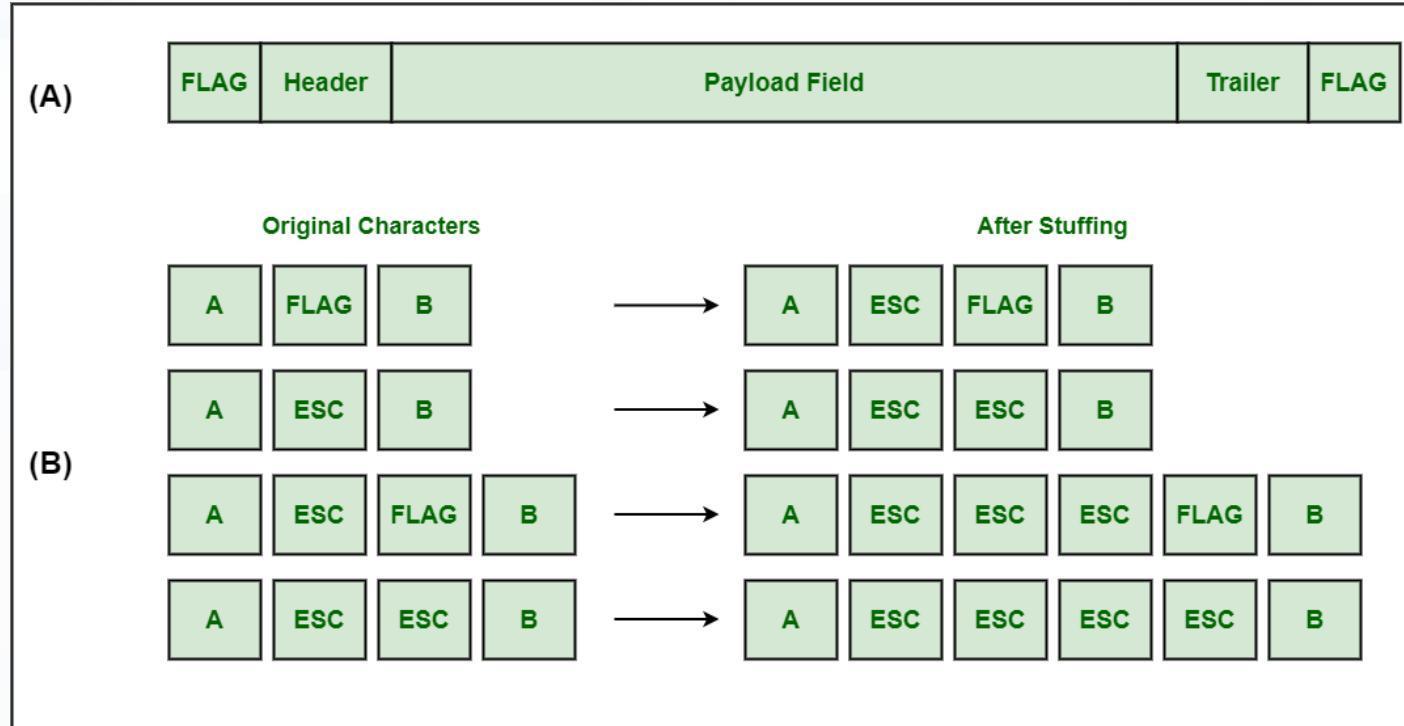
# L2: Znakově orientovaný přenos

- Základní jednotka kterou detekuji je 1 znak
- 1 znaků je tvořen N-bity, kde N může být různé dle konkrétního protokolu
  - Např 8, 7, 5 ...
- Začátek a konec rámce je uvozen speciálními znaky
  - Takzvané řídící znaky
- V případě detekce rámce dle označení začátku a konce potřebujeme dva řídící znaky
  - Začátek rámce - např STX nebo BOF
  - Konec rámce – např ETX EOF
- Rámcem pak vypadá při přenosu
  - **STX|HLAVIČKA|DATA|FCS|ETX**

# L2: Znakově orientovaný přenos - transparentnost přenosu

- Řídící znaku ( např STX A ETX ) nám umožní detekovat začátek a konec rámce
  - STX|HLAVIČKA|DATA|FCS|ETX
- Ale co když se vyskytnou v datech ?
  - STX|HLAVIČKA|DATA**STX**DATA|FCS|ETX
  - Nastává problém, protože nevíme zda je to začátek nové rámce nebo jen data!
- Je nutné zajistit „transparentnost“ přenosu – tedy jak bezpečně přenést i řídící znaky
- U znakově orientovaných přenosu používáme „escapování“
  - Tedy další řídící znak – např DLE – který říká, že to co následuje jsou JEN data a ne příkaz
  - STX|HLAVIČKA|DATA**DLE****STX**DATA|FCS|ETX
- Po vyřešení STX a ETX nám ale vznikl nový problém – jak přenést DLE ?
  - STX|HLAVIČKA|DATA**DLE****STX**DATA**DLE**DATA|FCS|ETX
- Vyřešíme stejně, tedy opět escapováním
  - STX|HLAVIČKA|DATA**DLE****STX**DATA**DLE****DLE**DATA|FCS|ETX
- DLE se vkládá při odeslání do dat a při příjmu se z nich zas na vstupu odebírá

# L2: Znakově orientovaný přenos - transparentnost přenosu - příklad



## A Character Stuffing

(A) A frame delimited by flag bytes  
(B) Four examples of byte sequences before and after byte stuffing

Zdroj: <https://www.geeksforgeeks.org/various-kind-of-framing-in-data-link-layer/>

# L2: Znakově orientovaný přenos příklad SLIP

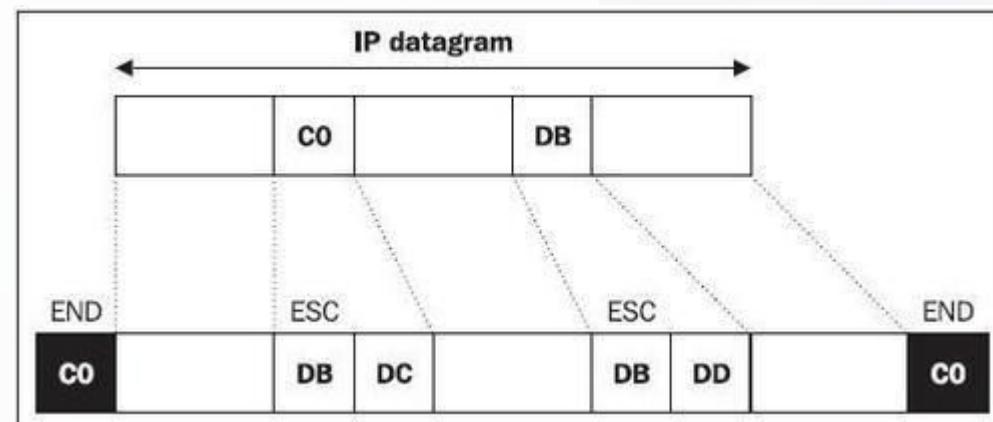
- Příkladem znakově orientovaného protokolu je např SLIP
  - Serial Line Internet Protocol
  - Protokol pro zapouzdření IP a pro jeho následný přenos po sériové lince

- Použité značky
  - Délka 8bit
  - Bez parity

Hex value	Dec Value	Oct Value	Abbreviation	Description
0xC0	192	300	END	Frame End
0xDB	219	333	ESC	Frame Escape
0xDC	220	334	ESC_END	Transposed Frame End
0xDD	221	335	ESC_ESC	Transposed Frame Escape

zdroj: [https://en.wikipedia.org/wiki/Serial\\_Line\\_Internet\\_Protocol](https://en.wikipedia.org/wiki/Serial_Line_Internet_Protocol)

- Příklad rámce
  - Rámcem „začneme“ značkou END
  - Pokud v datech máme ESC, pošleme ESC, ESC\_ESC
  - Pokud v datech máme END, pošleme ESC, ESC\_END
  - Pokud mám v datech ESC\_END nebo ESC\_ESC nic se neděje, protože k jejich aktivaci je třeba znak ESC před



zdroj: [https://subscription.packtpub.com/book/networking\\_and\\_servers](https://subscription.packtpub.com/book/networking_and_servers)

# L2: Bytově orientovaný přenos

- Bytově orientovaný protokol pracuje s pevně danou sekvencí bitů – bytem – jako znakově orientovaný protokol
  - Tedy se dá říct, že „znak“ má fixní délku 8 bitů
- Ale data neinterpretuje jako znaky, ale pracuje se sekvencí bitů
- Fakticky vždy máme jednotlivé bity – jde o interpretaci
  - Samostatné bity
  - Různě dlouhé sekvence bitů tvořící znak
  - Pevně daná sekvence bitů o velikosti 8 bitů – jeden byte
- Používá křídlovou značku – tedy sekvenci bytů – např **8x 01010101** pro Ethernet
- Stejná křídlová značka může být použita pro začátek i konec přenosu
- Je méně efektivní z hlediska využití přenosové kanálu než bitový přenos
  - Logicky, i pokud chci přenést menší data než 1 byte, přenese se 1 byte

# L2: Bytový přenos - transparentnost přenosu

- Stejně jako i bitového a znakového přenosu musí být i u bytového přenosu zajištěna transparentnost přenosu – tedy možnost přenosu křídlového bytu/flagu v datech
- Transparentnost se zajišťuje podobně jako v znakově orientovaném protokolu a tedy „escape“ bytem – v tomto případě je to celý byte
- Identicky i přenos „escape“ značky je, stejně jako u znakově orientovaného přenosu, řešen zdvojením „escape“ bytu
- Příklad
  - Flag 01111110, escape 11111111, data 01111110|011001100|11111111
  - Přenos bude
  - 01111110|**11111111**|01111110|011001100|**11111111**|11111111|01111110

# L2: Bytově orientovaný přenos příklad Ethernet

- Příkladem L2 bytově orientovaného protokolu je Ethernet
  - Jeden z nejpoužívanějších protokolů dneška
- Existuje dnes více variant
  - **Ethernet II / IEEE802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications / 802.2 SNAP / 802.2 LLC**
  - Základní rámec je velice podobný, liší se v drobnostech
    - Preamble
      - 01010101 – slouží pro synchronizaci
      - SOF – Start Of Frame 01010111 – křídlová značka
    - Cílová adresa: MAC
    - Zdrojová adresa: MAC
    - Délka / Typ
      - IEEE 802.3 – délka dat ( max 1500 )
      - Ethernet II – type rámce ( > 1500 )
    - 802.2 Header + Data / Data
      - IEEE 802.3 - 802.2 Header + Data
      - Ethernet II - Data
    - FCS: zabezpečení

IEEE 802.3								
Preamble	SOF	Destination Address	Source Address	Length	802.2 Header	DATA	FCS	
7	1	6	6	2	46-1500		4	

Ethernet II					
Preamble	Destination Address	Source Address	Type	DATA	FCS
8	6	6	2	46-1500	4

# L2: Chyb v přenosu

- V reálném přenosu mohou vznikat chyby téměř vždy
- Zdroje chyb
  - Bílý – tepelný šum
    - Pohyb elektronu / útlum signálu – ten prostě je a nic s tím neuděláme
    - Je trvale přítomen
  - Impulzní šum - elektromagnetické rušení
    - Typicky nějaká interference například se silovým vedením, při zapnutí spotřebiče atd
    - Vadí, ale není trvalý
  - De-synchronizace vysílače a přijímače
    - Dojde k de-synchronizaci hodin a vysílač vyšle třetí bit, ale přijímač jej přijme jako druhý nebo čtvrtý
  - Výpadek celého bloku dat
    - Dočasný výpadek spojení – například mokré listí ve WiFi „cestě“

# L2: Spolehlivý a nespolehlivý přenos

- !! POZOR !! - Spolehlivost je možné řešit na více vrstvách, zde se bavíme o L2, ale stejně tak jej řeší TCP na L4
- Spolehlivý přenos
  - Typicky z pohledu vyšší vrstvy, které poskytujeme služby
    - Jednoduše řečeno, že se můžu na přenos spolehnout a je mi jedno jak je zařízen
  - Potřebujeme vědět, že
    - Data se přenesla všechna – nic nechybí
    - Data se přenesla správně – všechny 0 a 1 jsou na svém místě
    - Data se přenesla ve správném pořadí – tedy všechny rámce jsou na svém místě
  - Pokud výše uvedené není splněno, potřebujeme to mít možnost nějak napravit
    - Samoopravné kódy, re-transmit
  - Požadavek je to logický, ale pochopitelně drahý, protože když se něco nepovede, musím to nějak řešit
- Nespolehlivý přenos
  - Můžu, ale nemusím vědět, že se něco nepovedlo
  - Můžu, ale nemusím zjištěný problém nějak řešit
    - „Já sice vím, že je problém, ale také vím, že na výsledek pro který se přenos provádí to nebude mít vliv a tedy to neřeším, protože bych zdržoval“
  - Typicky u služeb, kde výpadek části dat nevadí
    - Například hlasové a multimedialní přenosy – výpadek několika málo rámce za sebou nebude ve výsledku vůbec patrný

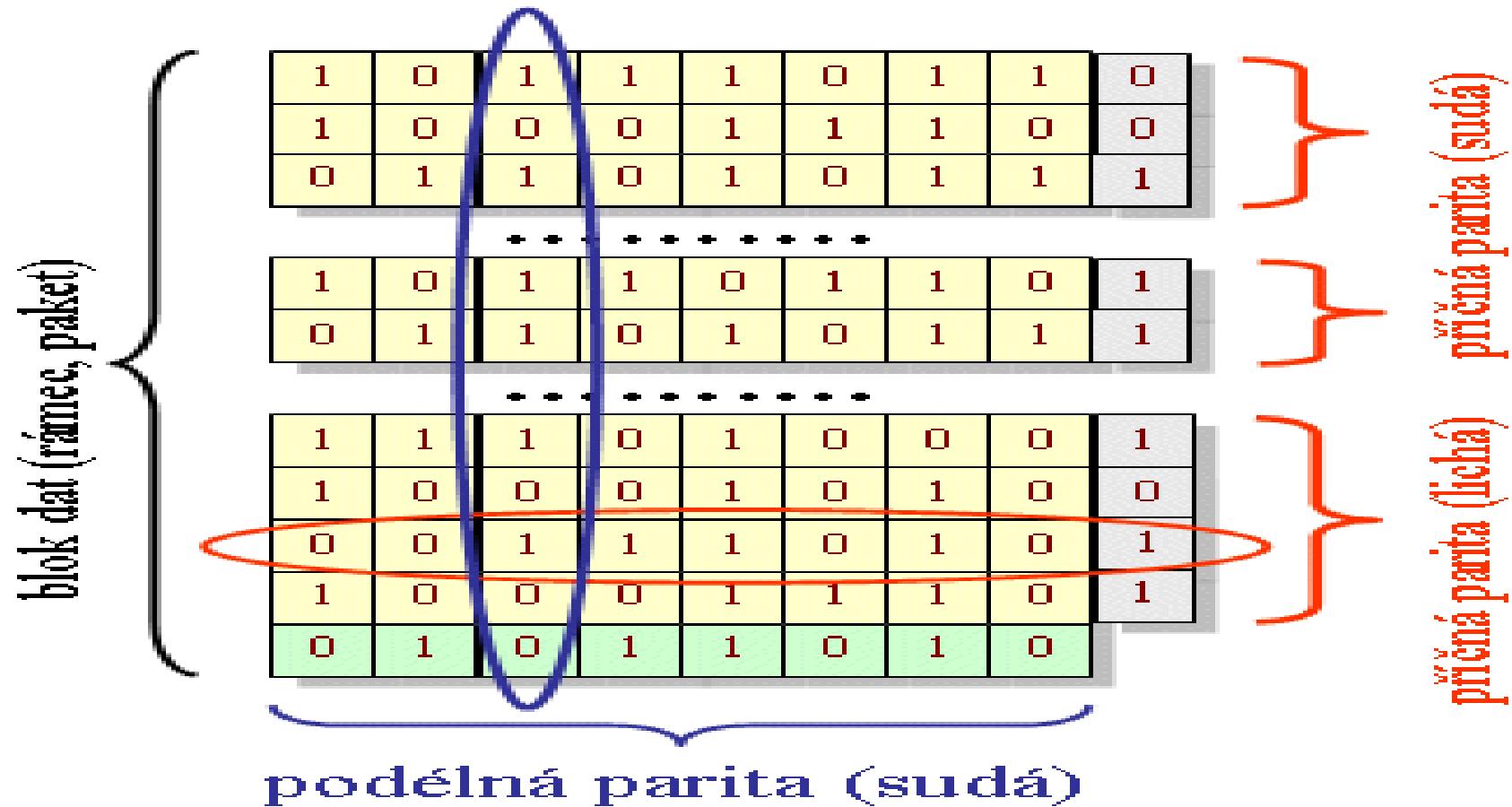
# L2: Detekce chyb

- Aby bylo možné realizovat spolehlivý přenos, je nutné se o chybě dozvědět
  - => Detekce chyb
- Detekce chyb je opět možné řešit na více úrovních
  - Detekce výpadku celého bloku / rámce
    - Zjistím tak, že mi nesedí čísla rámců – nějaký chybí nebo přišel ve špatném pořadí
    - V hlavičce rámce musím zavést číslování – ne každý protokol to tak má – viz další přednáška
  - Detekce chyby v jednom rámci / v jednotlivých bitech
    - Zjistím za pomocí přidaných bitů a kontrolních mechanizmů jako jsou:
      - Parita
      - Kontrolní součty / check sumy
      - Cyklické kódy / CRC
    - Výsledek kontrolního mechanizmu je připojen na konec dat a přenášen společně s nimi s slouží jako podklad pro kontrolu správnosti přenosu
- Opakování / souvislost
  - Podobně jako u arytmického přenosu zde vstupuje do hry využití přenosového kanálu
    - Přenáším data, které přenášet nechci, ale musím => režie přenosu,  $n = N/M$
  - Abych minimalizoval vliv režie, chci co nejdelší blok dat – na L2 rámec, ALE
    - Každý jeden bit se přenese úspěšně s pravděpodobností  $p_1$ , respektive s chybou  $q=1-p_1$
    - Pravděpodobnost úspěšného přenosu  $N$  bitů je  $P_N = p_1^N$
    - Pokud víme s jakou pravděpodobností chceme přenos realizovat a jaké je pravděpodobnost chyby, je maximální délka rámce  $N = \log_{p_1}(P_N)$  – CELÝCH bitů – nezaokrouhlujeme, ale ořezáváme – logicky nemůžeme přenést „půl bit“

# L2: Zabezpečení přenosu: Parita

- O paritě už jsme se zmiňovali u arytmického přenosu, kde byla jako volitelná součást přenosu na L2
  - Jeden z důvodů proč v TCP/IP modulu je L1 a L2 spojené => ne vždy jde přesně rozdělit
- Parita je přidaný 1 bit, kde hodnota je taková aby výsledný počet 1 byl:
  - Sudý: pro  $1110|P_s \Rightarrow 1110|1$  ( doplním 1 na  $4 \times 1 \Rightarrow$  sudý počet )
  - Lichý: pro  $1110|P_L \Rightarrow 1110|0$  ( doplním 0, protože v datech mám  $3 \times 1 \Rightarrow$  lichý počet )
- Základní, tedy sudá nebo lichá parita, NEMUSÍ odhalit chybu vždy
  - Problém u násobných chyb – pokud se mi změny dva bity, chyba se sečte, parita bude v pořádku, ale data ne => DETEKCE není na 100%
    - Následně se musí řešit i na vyšších vrstvách – dle potřeb protokolu / služby
- Další možnosti jsou složené parity pro bloky dat
  - Podélná parita – přidám paritní bit pro každý sloupec v bloku
  - Příčná parita - přidám paritní bit pro každý řádek ( slovo ) v datech
- Podélná i příčná parita mohou být jak sudé tak liché tak jejich kombinace
  - Záleží na autorovi přenosu jaké nastaví parametry
- Existuje i „jedničková“ a „nulová“ parita – tedy paritní bit je vždy 1 nebo 0
  - Tento druh „parity“ nemá žádný zabezpečovací význam

# L2: Zabezpečení přenosu: Parita příklad



## L2: Zabezpečení přenosu: Parita - detekce chyby - příklad

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
<hr/>					
1	0	1	0	1	0

*no errors*

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
<hr/>					
1	0	1	0	1	0

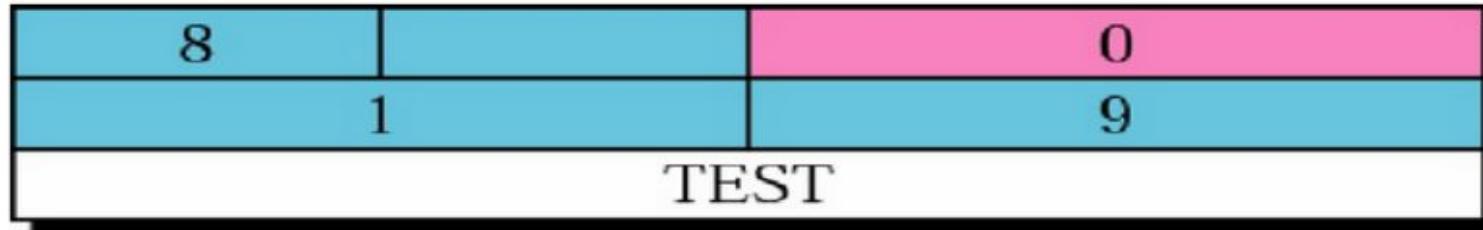
parity error

parity error

# L2: Zabezpečení přenosu: Kontrolní součet / Checksum

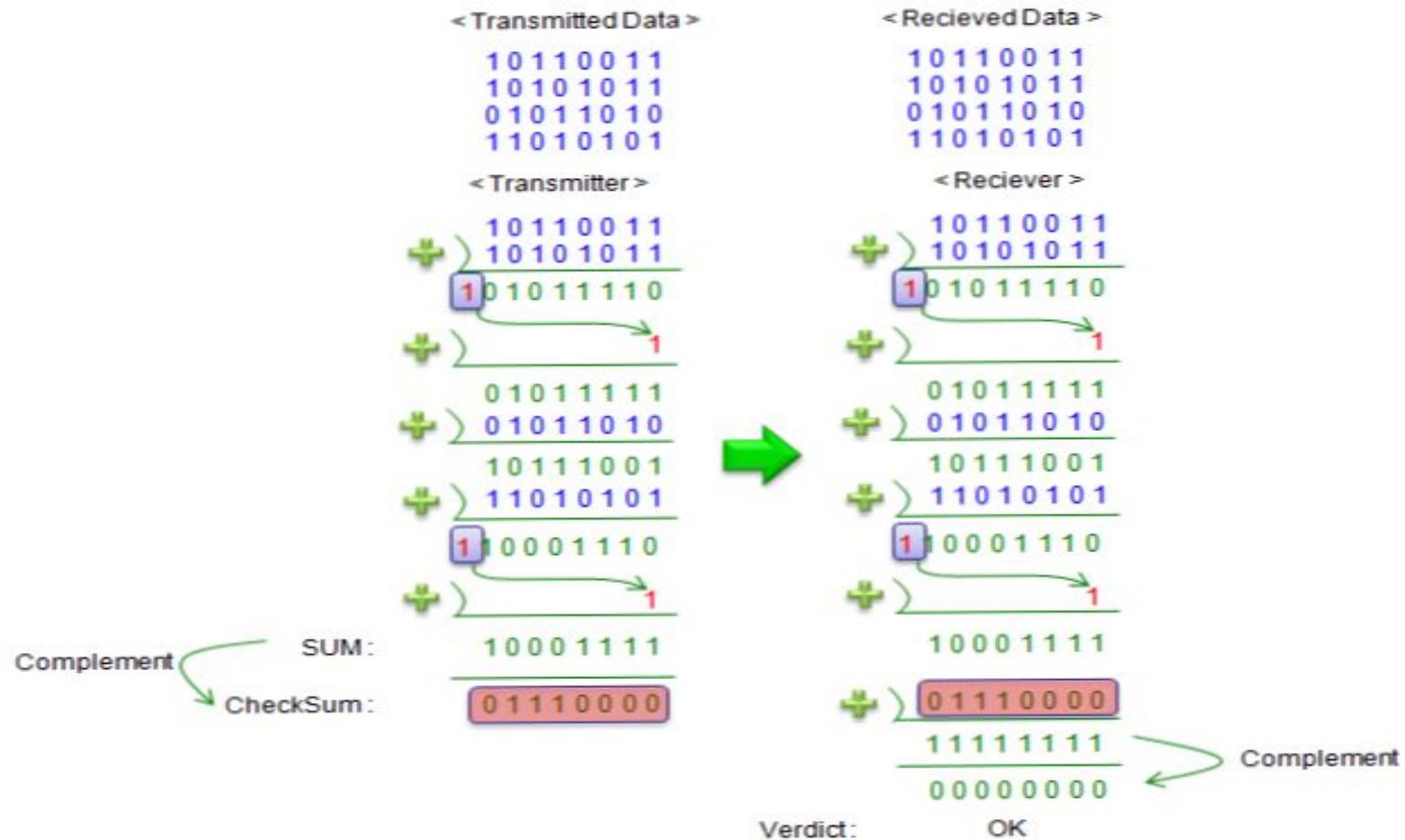
- Jednoduchá parita není příliš účinná, protože neodhalí násobné chyby
  - Ale přesto má smysl, protože se snadno realizuje a nepotřebuje „mnoho“ data navíc
- Podélná a příčná parita už nají účinnost lepší – odhalí více chyb, ALE
  - Spotřebuje více bitů => klesá efektivita využití přenosového kanálu
  - Pro podélnou paritu potřebujeme držet celý blok dat aby bylo možné ji spočítat
    - Paměťové i časově nepříjemné
- Kontrolní součet se snaží řešit výše uvedené problémy
  - Data jsou interpretována jako „slova“ - tedy jednotlivé byty - které mohou být znak nebo číslo
  - Jednotlivá slova, tak jak probíhá přenos, se postupně sčítají
    - Mohu dělat průběžně, takže nepotřebuji držet celý blok dat v paměti, ale vždy jen jedno slovo / byty
      - Může jít o sčítání nebo o XOR
  - Výsledný součet se připojí k datům a odešle společně s nimi a na straně příjemce se opět provede kontrolní součet ( !! POUZE DAT, kontrolní součet do výpočtu nevstupuje ) a porovná s přeneseným kontrolním součtem
    - Přesněji se nepřenáší přímo výsledek, ale jen zbytek po modulo N, kde N je délka slova ( pro 1 byte je to 8 )
    - Tím, že přenášíme zbytek po modulo N, nebude ten zbytek nikdy delší než N => maximální pevná délka

# L2: Zabezpečení přenosu: Kontrolní součet / Checksum příklad



8 & 0	→	00001000	00000000
0	→	00000000	00000000
1	→	00000000	00000001
9	→	00000000	00001001
T & E	→	01010100	01000101
S & T	→	01010011	01010100
Sum	→	10101111	10100011
Checksum	→	<b>01010000</b>	<b>01011100</b> —

# L2: Zabezpečení přenosu: Kontrolní součet / Checksum - příklad

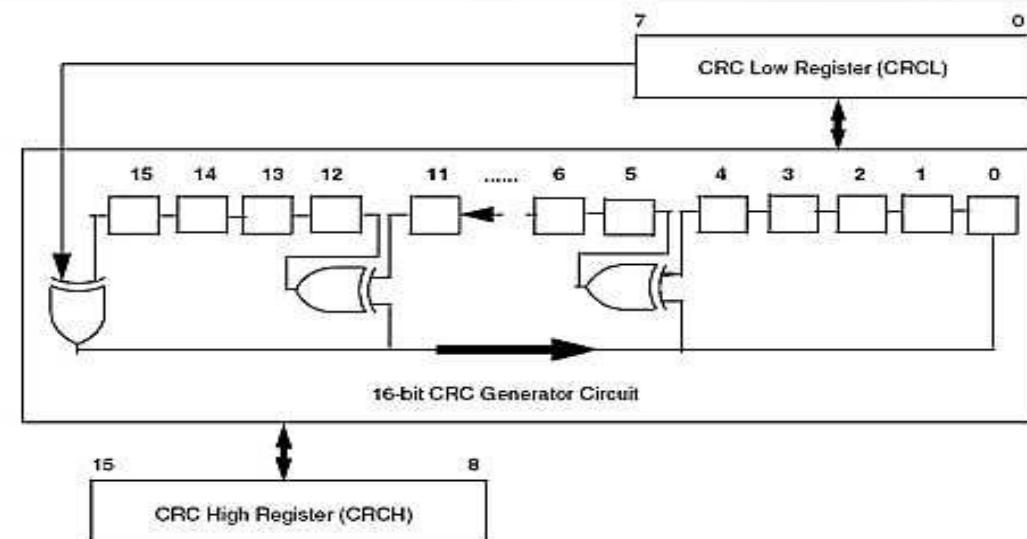


# L2: Zabezpečení přenosu: CRC

- CRC - Cyclic Redundancy Check - cyklické zabezpečovací kódy / polynomy
- Ani parita ani kontrolní součet nemají dostatečnou přesnost v detekci chyb
- CRC představuje další možnost v zabezpečení a detekci chyb v přenosu s výrazně lepšími výsledky než parita nebo checksuma
- CRC interpretuje posloupnost dat ( 0 a 1 ) - „slovo“ - jako polynom N-tého řádu
  - $101 \Rightarrow 1^*x^2 + 0^*x^1 + 1^*x^0 \Rightarrow x^2 + 1$
- Princip zabezpečení vychází z dělení polynomů:
  - Máme data, která chceme vysílat  $M(x)$
  - Zvolíme zabezpečující polynom  $G(x)$  - někdy také označován jako generující polynom
    - Například pro CRC-16:  $x^{16}+x^{15}+x^2+1$
  - Provedeme dělení  $M(x)/G(x) \Rightarrow R(x)$ 
    - Před dělením přidáme k  $M(x)$  počet nulových bitů odpovídající stupni zabezpečujícího polynomu -1
    - Kde  $R(x)$  není výsledek dělení, ale zbytek po dělení
    - Zbytek po dělení nikdy nebude delší než je řád generujícího polynomu
      - $\Rightarrow$  máme pevnou délku zabezpečení
    - Zabezpečující polynom může být výrazně kratší než přenášená data
      - $\Rightarrow$  máme dobré využití přenosového kanálu
  - Přenášet budeme  $T(x)$ , kde  $T(x) = M(x)+R(x)$  ( prosté přidání zbytku po dělení za data )
  - Po přijetí dat na straně přijímače provedeme kontrolní dělení  $T(x)/G(x)$ 
    - Pokud zbytek po kontrolním dělení není nula  $\Rightarrow$  nastala chyba
    - Pokud zbytek po kontrolním dělení je nula  $\Rightarrow$  přenos se podařil správně

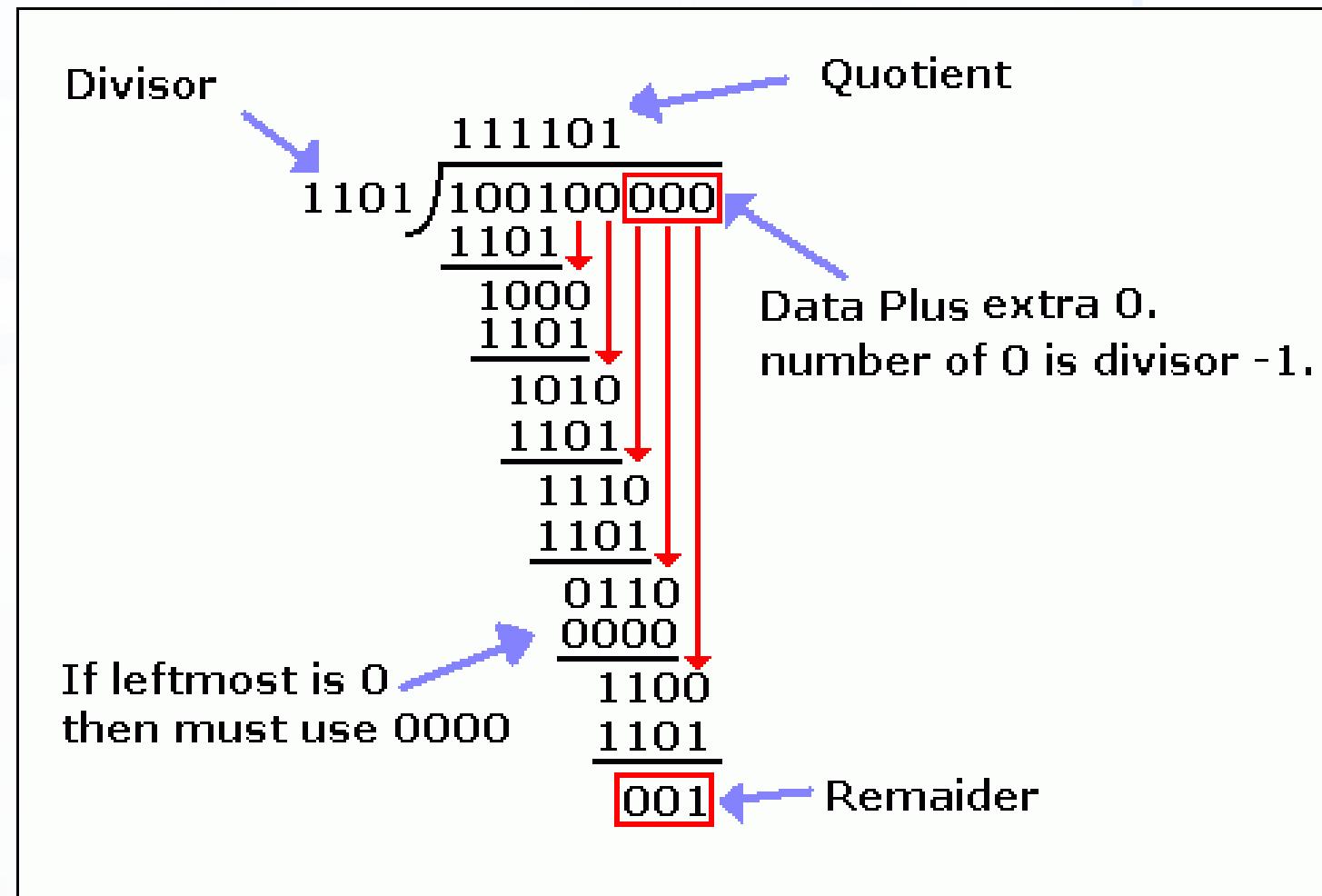
# L2: Zabezpečení přenosu: CRC II.

- Schopnost detekce chyb je u CRC velice vysoká
  - Dokáže detektovat všechny shluky chyb o velikosti  $>N+1$  s pravděpodobností 99.9999998% ( pro CRC-32 např )
    - Tedy chyby, které jsou delší než je zabezpečující polynom
    - To je výhodné, protože na úrovni rámců je častější shluk chyb než samostatná chyba
- Výpočet CRC je jednoduše realizovatelný na úrovni HW
  - Soustava XOR-hradel a posuvných registrů
  - Zásadní a složitá je vhodná volba zabezpečujícího polynomu



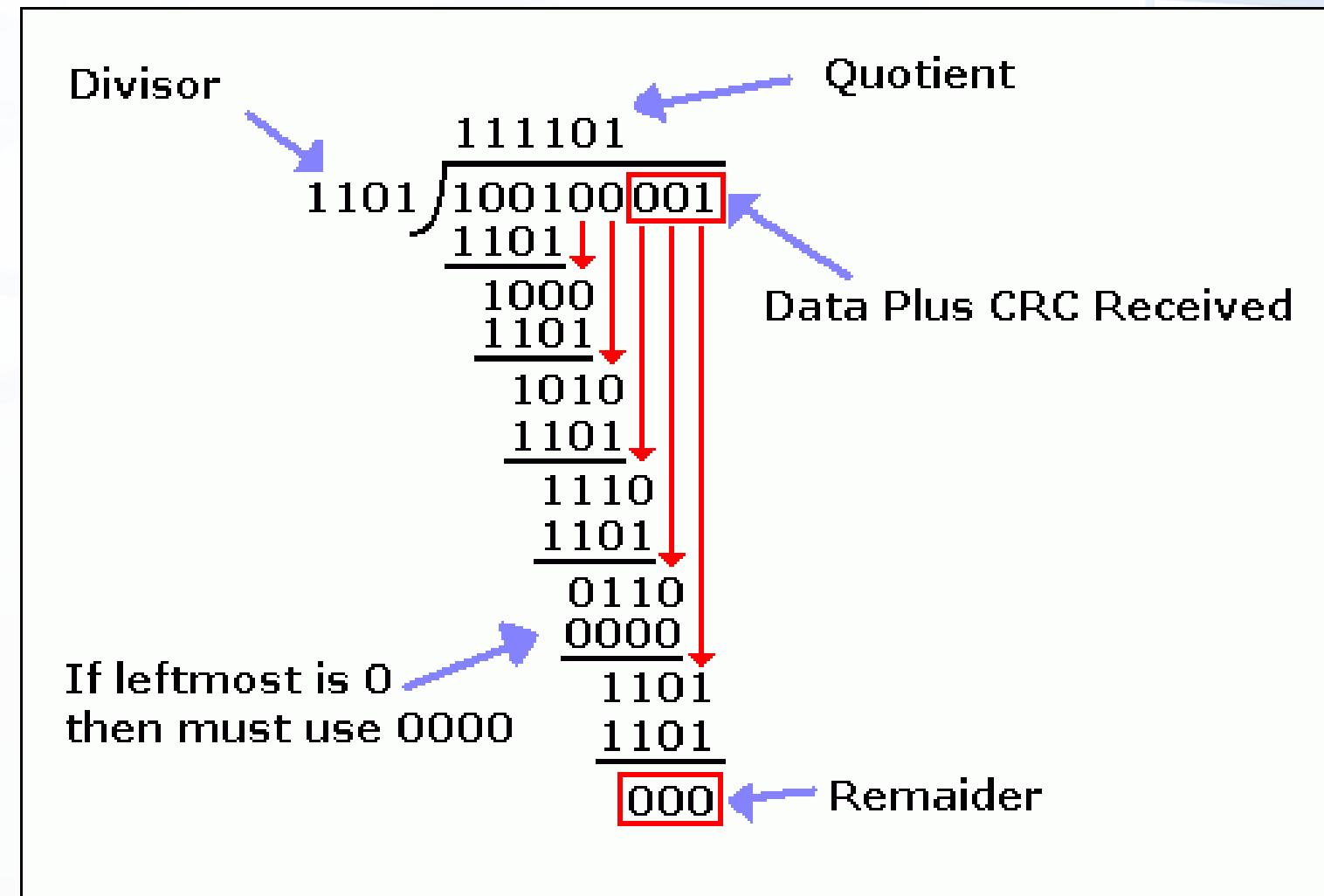
# L2: Zabezpečení přenosu: CRC příklad výpočtu

- Máme data  $M(x)=111101$
- Máme zabezpečující polynom  $G(x) = 1101$
- Za  $M(x)$  přidáme nulové byty
  - Počet „řad“  $G(x)-1 = 3$
- Provedeme dělení, kde dostaneme zbytek  $R(x) = 001$
- Přenášená zpráva bude  $T(x) = 111101|001$



# L2: Zabezpečení přenosu: CRC příklad kontroly

- Máme zprávu  $T(x)=111101|001$
- Máme zabezpečující polynom  $G(x) = 1101$
- Provedeme kontrolní dělení, kde  $R(x) = 000$ 
  - => Přenos se realizoval správně



# L1/L2: Opakování: Využití kapacity přenosového kanálu

- U arytického přenosu jsme měli 1 paritní bit a start a stop bit a už to negativně ovlivnilo využití kapacity přenosového kanálu, co potom CRC na např 15 bitech ?
- Příklad 1: mějme datový rámec o délce 32 bitů a zabezpečující polynom o délce 15 bitů, jaké bude využití přenosového kanálu ?
  - $n = N/M$ ,  $N = 32$  bitů,  $M = 32 + 15 = 47$ ,  $n = 32/47 = 0,68 \sim 68\% \Rightarrow$  bída
- Příklad 2: mějme datový rámec o délce 1500 bitů a zabezpečující polynom o délce 15 bitů, jaké bude využití přenosového kanálu ?
  - $n = N/M$ ,  $N = 1500$  bitů,  $M = 1500 + 15 = 1515$ ,  $n = 1500/1515 = 0,99009901 \sim 99\% \Rightarrow$  lepší

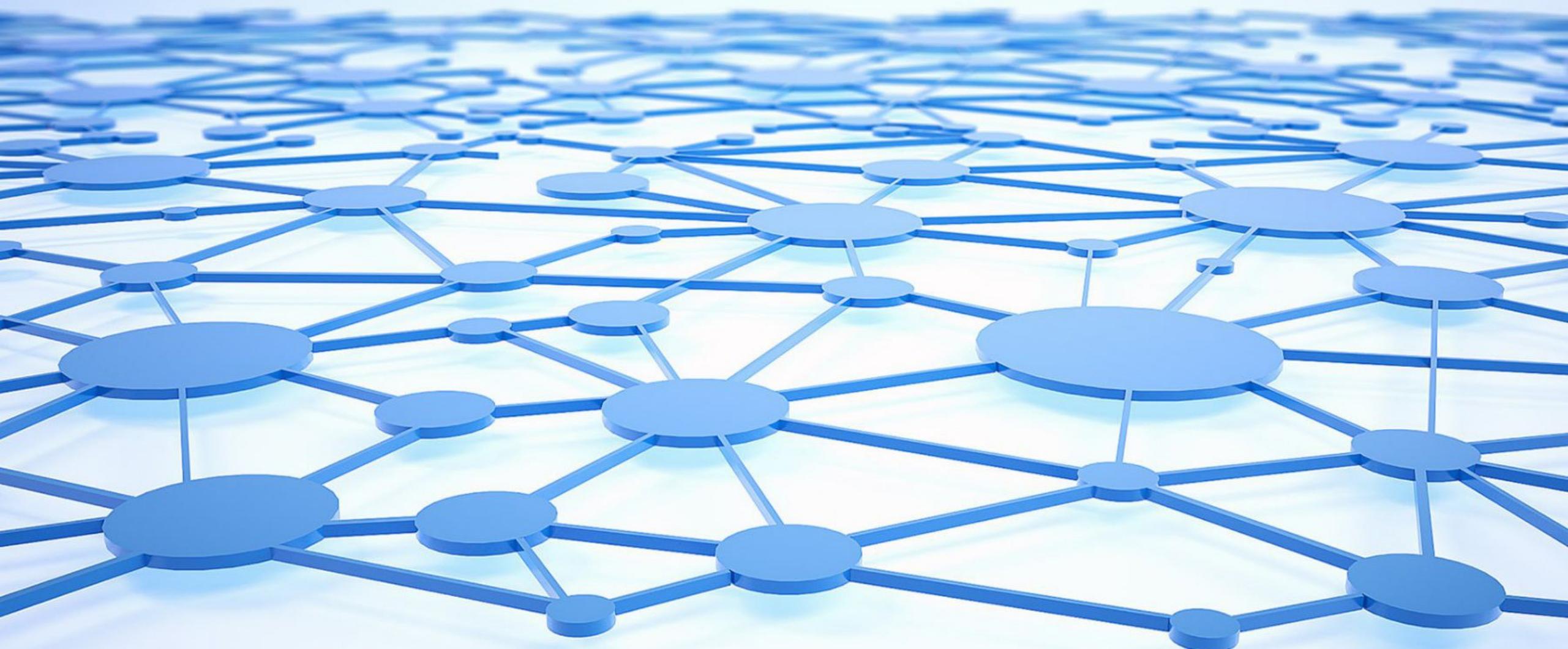
# L1/L2: Opakování: Pravděpodobnost úspěšnosti přenosu

- Příklad 3: mějme datový rámec o délce 1500 bitů a zabezpečující polynom o délce 15 bitů, pravděpodobnost chyby v 1 bitu 99.999% - jaká bude pravděpodobnost úspěšného přenosu ?
  - $P_N = p_1^N$ ,  $p_1 = 0.99999$ ,  $N = 1515$ ,  $P_N = 0.99999^{1515} = 0,98496 \sim 98\%$
- Příklad 4: Jak se změní pravděpodobnost spěšného přenosu, pokud prodloužíme délku datového rámce na 15000 bitů ?
  - $P_N = p_1^N$ ,  $p_1 = 0.99999$ ,  $N = 15015$ ,  $P_N = 0.99999^{15015} = 0,86058 \sim 86\%$
- => nemohu délku rámce zvyšovat bez následků
- => je nutné hledat vhodný kompromis bez využitím kapacity přenosového kanálu a pravděpodobností chyb/úspěšnosti přenosu

# Úvod do počítačových sítí

Přednáška 6 ( 2025/2026 )

ver. 2025-10-15-01



# Řešení chyb v přenosech

- Příchozí rámec je kontrolován a kontrola selže, co dále ?
  - Kontrola pomocí parity, checksumy, crc ...
- Na chybu je třeba reagovat primárně tím, že poškozený rámec nedám k dalšímu zpracování vyšší vrstvě
  - Proč bych to také dělal když už vím, že je špatně a tedy že výsledek správný nebude, ale může vzniknout více chyb
- Možností řešení je více a záleží na protokolu a použití
  - Nemusíme dělat nic
  - Můžeme zkoušet chybu opravit ( FEC - Forward Error Correction )
  - Můžeme se pokusit přenos opakovat ( ARQ - Automatic Request Query )

# Zahazování poškozených rámců

- Pokud zjistíme, že rámec není v pořádku nemusíme v některých případech „dělat nic“
- „Dělat nic“ v reálu znamená vynechat dvě akce
  - Předat data z rámce L3 vrstvě – NEDĚLÁM
  - Dávat odesilateli zprávu, že je něco špatně
- Pochopitelně nejjednodušší řešení, ale jde použít jen někde
- Typicky v multimedialních streamech
  - Výpadek jednoho či několika málo rámců nevadí
    - Uživatel výpadek nezaznamená a nebo jen nepatrнě – nepoškozuje to úplně funkčnost vysílání
    - Řešení by bylo „časově drahé“ a tedy viditelnější než když neudělím nic

# Oprava poškozených rámců

- FEC - Forward Error Correction
- Oprava může být, po nic nedělání, nejrychlejší cesta jak se chybou v přenosu vypořádat
  - Nemusím posílat info odesílateli, že se něco nepovedlo a pak čekat na nová data
- Oprava dat není možné vždy / u všech typů přenosu
- Pro opravu potřebuji mít:
  - Omezenou množinu dat vhodně kombinovanou se zabezpečením
  - Přenášet jakákoliv data, ale vhodně upravená - „kódovaná“
    - Pozor zde se nejedná o kódování RZ, Manchester atd - to bylo na L1
    - Název je stejný, ale smysl je posílat data tak, aby se v nich dala nalezená chyba opravit
- Výskyt malého počtu chyb
  - Pokud je chyb hodně většinou není oprava vůbec možná
  - Teoreticky by možná být mohla, ale počet násobných zabezpečení by zabral většinu kapacity přenosového kanálu a to už pak nedává smysl

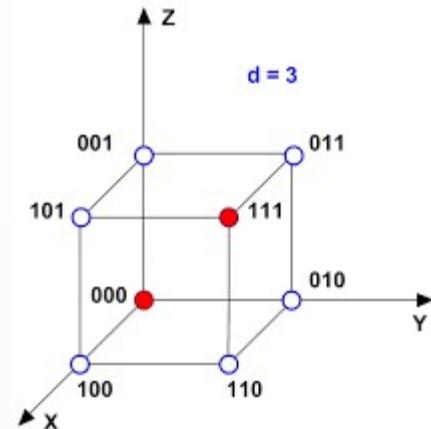
# Hamingova vzdálenost

- Nemůžou se přenášet úplně libovolná data, ale jen vybraná
  - Např jen 010 a 101
    - Pro úplně libovolný přenos to nedává smysl, ale pro speciální - například řídící operace - ano
  - Pokud budu mít pevně danou množinu přenášených „slov“ mohu snáze určit chybu a zároveň ji mohu opravit
    - Samozřejmě za předpokladu, že se nejedná o násobné chyby
- Možnost najít v datech chybu a případně jí opravit je závislá na „rozdílnosti!“ dat
  - Pokud přenáším 1111 a 0000 tak 1110 je zjevně chyba „pravděpodobně“ se mělo jednat o 1111
  - Pokud přenáším 1111 a 1110 tak 1110 může být ok, ale také to může být chyba v prvním slově
- Z přenášených dat můžeme určit Hammingovu vzdálenost
  - Jedná se o počet bitů, ve kterých se přenášená slova liší
- Označuje se  $d_H$
- Minimální Hammingova vzdálenost je nejmenší Hammingova vzdálenost v celém kódu
  - Tedy nejmenší hodnota pro všechny možné dvojice

# Hamingova vzdálenost: Metody určení

- „Ručně“ - odpočtením bitů
  - 00 a 01 =>  $d_H = 1$
  - Problém u delších slov, problém pro strojové zpracování
- Pomocí XOR pro každou dvojici
  - 000 a 001 =>

000
001
-----
$0+0+1 = d_H = 1$
- „Přenosem kódu na krychli“
  - Jednotlivé vrcholy definují jednotlivá slova
  - Vzdálenost vrcholů určuje vzdálenost dvou slov
    - Kolika hranami musím projít, abych se dostal z jednoho vrcholu na druhý



# Hamingova vzdálenost příklad

- **Příklad 1:** Přenášíme data: 1111, 0000, 1010, určete minimální Hammingovu vzdálenost
  - Nejprve určíme vzdálenost všech kombinací slov:
    - 1111 a 0000 => 4 ( 4 bity jsou různé )
    - 1111 a 1010 => 2 ( 2 bity jsou různé )
    - 0000 a 1010 => 2 ( 2 bity jsou různé )
  - Minimální Hammingova vzdálenost ( tedy nejmenší nalezená vzdálenost ) je 2,  $d_{H\min} = 2$ 
    - Nejméně ve dvou bitech se všech slova liší
- **Příklad 2:** Určete minimální Hammingovu vzdálenost pro kód, který přenáší libovolné kombinace bitů o délce znaku 2 ( 00,11,01,10 )

00	00	00	11	11	01
11	01	10	01	10	10
---	---	---	---	---	---
1+1=2	0+1=1	1+0=1	1+0=1	0+1=1	1+1=2

- $d_{H\min}(00,11,10,01) = 1$ 
  - ( je to min z 2,1,1,1,2 )

# Hamingova vzdálenost a detekce a oprava chyb

- Tím, že Hammingova vzdálenost vypovídá o vlastnostech kódu, může vypovídat o jeho možnostech detekce a případně korekce chyb
  - Logicky, čím větší rozdíly v povolených slovech, tím snadnější poznání, že je něco špatně a „snad“ i jak by to mělo být dobré
    - „Snad“ proto, že při násobné chybě nemusí být oprava možná
- Detekce chyb
  - $d_{H\min} \Rightarrow n+1$                                    $\Rightarrow n \leq d_{H\min} - 1$
  - Jsem schopen detekovat nejvíce chyb jako je rovno  $d_{H\min} - 1$
- Oprava chyb
  - $d_{H\min} \Rightarrow 2n+1 \Rightarrow n \leq (d_{H\min} - 1)/2$
  - Jsem schopen opravit maximálně kolik chyb, kolik odpovídá  $(d_{H\min} - 1)/2$

# Hamingova vzdálenost a detekce a oprava chyb: příklad

- Příklad 1: Kolik chyb jsem schopen detektovat, pokud přenáším jen slova 0001, 1111, 1010
  - Detekce chyb  $d_{Hmin} \Rightarrow n+1$   $\Rightarrow n \leq d_{Hmin} - 1$
  - $D_{Hmin} = \begin{array}{cccc} 0001 & 0001 & 1111 \\ 1010 & 1111 & 1010 \\ \hline 1+0+1+1=3 & 1+1+1+0=3 & 0+1+0+1=2 \Rightarrow d_{Hmin}=2 \end{array}$
  - $n \leq d_{Hmin} - 1$ ,  $n \leq 2-1$ ;  $n \leq 1 \Rightarrow$  Jsem schopen detektovat 1 chybu
  - Pokud mám jen jednu chybu poznám to, protože změna jednoho bitu nevede ke změně na jiné platné slovo
- Příklad 2: Kolik chyb jsem schopen opravit, pokud přenáším jen slova 0001, 1111, 1010
  - Oprava chyb
    - $d_{Hmin} \Rightarrow 2n+1 \Rightarrow n \leq (d_{Hmin}-1)/2$
    - $d_{Hmin} = 2$
    - $n \leq (d_{Hmin}-1)/2$ ;  $n \leq (2-1)/2$ ;  $n \leq 0,5 \Rightarrow$  nejsem schopen opravit žádnou chybu
      - Logicky, protože pracuji na úrovni celých bitů a vyšlo mi „půl bitu“
      - Opravit nemohu nic, protože pro 1011 nepoznám, zda správně bylo 1111 nebo 1010

# Korekce chyb: Samoopravné kódy

- Vhodným způsobem upravím přenášená data, aby obsahovala násobnou možnost detekce chyb a tím i umožnila opravu
  - Upravím zde znamená „doplním o zabezpečovací bity“
  - Úprava jde samozřejmě na úkor využití kapacity přenosového kanálu
    - Logicky, čím více bitů na zabezpečení, tím méně prostoru pro bity datové
  - Typicky se jedná o násobné zabezpečení jednoho bitu
    - Nejjednodušším příkladem může být kombinace podélné a příčné parity, protože vím, že chyba je v konkrétním řádku a zároveň sloupci a tím pádem i vím jaká správní hodnota tam má být
    - Problém násobných chyb řeší jen částečně – násobné chyby se mohou „vymaskovat“
- Příkladem mohou být
  - Hammingovo kódování
    - ECC RAM, bezdrátová komunikace, satelitní přenosy
  - BCH kódy
    - SSD, Flash, CD/DVD, satelitní přenosy, CAN

# Korekce chyb: Hammingovo kódování

- Jedná se speciální příklad příklad lineárních dvojkových kodů ( $n,k$ )
- Doplňuje násobné zabezpečovací bity
- Kódy  $(3,1)$ ,  $(7,4)$ ,  $(15,11)$ ,  $(31,26)$ , ...
  - Často použitá varianta je  $(7,4)$ 
    - $d_{Hmin} = 3$
    - Dvě chyby detekuje
    - Jednu chybu opravuje
- $n$  - délka slova  $r$ - paritní bity  $k$  - informační bity
- Princip tvorby
  - Paritní bit jsou na pozicích druhých mocnin  $(1, 2, 4, 8, 16, \dots)$
  - Informační bity jsou na ostatních pozicích  $(3, 5, 6, 7, 9, 10, 11, \dots)$
  - Paritní bit se vypočítá z **některých** bitů informačního slova. Pozice informačních bitů, které se vyneschávají udává pozice paritního bitu
  - $P_1 P_2 I_1 P_3 I_2 I_3 I_4$
  - $P_1 = I_1 + I_2 + I_4$  ( začínám na prvním bitu a jeden zkонтroluji, jeden přeskočím, jeden zkонтroluji ... )
  - $P_2 = I_1 + I_3 + I_4$  ( začínám na druhém a dva zkонтroluji, dva přeskočím, dva zkонтroluji .... )
  - $P_3 = I_2 + I_3 + I_4$  ( začínám na čtvrtém bitu a čtyři zkонтroluji, čtyři přeskočím, čtyři zkontroluji, .... )
- Tvořím s - syndrom, který slouží ke kontrole po přenosu a který musí být nulový
  - $S_1 = P_1 + I_1 + I_2 + I_4 = 0$
  - $S_2 = P_2 + I_1 + I_3 + I_4 = 0$
  - $S_3 = P_3 + I_2 + I_3 + I_4 = 0$

# Korekce chyb: Další kódování

- Rozšířené Hammingovo kódování
  - Rozšiřuje základní Hammingův kód o přidání paritního sloupce
  - Např z (7,4) vznikne (8,4)
    - 4 informační a 4 zabezpečovací
    - $d_H = 4$ 
      - Tři chyby dokáže detekovat, dvě chyby **bezpečně** detekuje
      - Jedno chybu opraví
- BCH kódy
  - Využívají CRC - cyklický
  - Nevkládá kontrolní byty mezi informační, ale připojuje je na konec zprávy
  - Používají se např při satelitních přenosech
    - Přidávám sice hodně informace navíc, ale zpoždění v přenosu ( RTT ) je tak veliké, že se to vyplatí

# Korekce chyb: Opakování přenosu

- ARQ - Automatic Request Query
  - Pokud detekuju chybu a
    - Nejsem schopen ji opravit ( Hammingovy kódy, BCH, ... )
    - Data potřebuji celá, protože se nejedná např o multimediální přenos
  - musím řešit opakováný přenos
  - Pro opakování přenosu je třeba dát odesílateli info, že je třeba data poslat znovu
  - Zavedeme potvrzování a timeout
  - Potvrzování
    - Potvrzení má typicky formu informačního rámce
      - Jsou i výjimky, kdy se potvrzení přibalují k datům, v případě duplexních kontinuálních přenosu při vysoké míře aktivity v komunikaci
    - Kladné - potvrzuji správně přijetí každého rámce ( ACK )
    - Záporné - posílám info v případě, že jsem detekoval chybu ( NACK )
    - Kladné i záporné - kombinace obou případů
  - Timeout
    - Nejen přenos, ale i doručení potvrzení může selhat
    - Řeší problém nekonečného čekání
      - Čekám X dlouho a následně přenos prohlásím za neúspěšný a mohu jej zopakovat

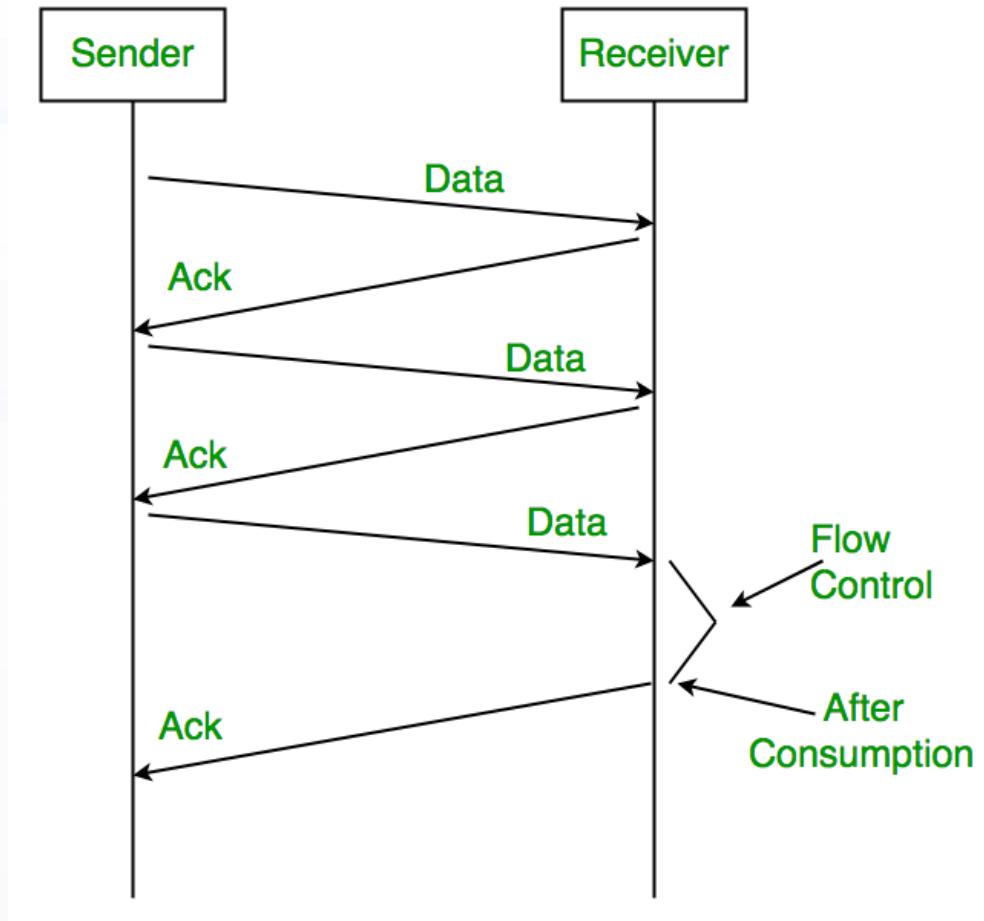
# Simplexní protokol bez omezení

- Simplexní pro jednoduchost
  - Data( myšleno co chci posílat ) posílám jen jedním směrem
- Bez omezení – posílám data včetně zabezpečení, ale nestarám se o to jak to dopadne
  - Nepoužívám potvrzování
  - Nepoužívám samoopravné kódy
- V běžných provozech na L2 jen omezené využití
  - Pro představu se může jednat o „alternativu“ UDP(L4), ale na L2
- Není možné použít pro všechny typy přenosů, ale jen
  - Tam kde výpadek části dat nevadí
  - O konzistenci dat se starají až vyšší vrstvy
    - Ale to obecně nechceme, protože je to časově náročnější

# Simplexní protokol Stop & Wait

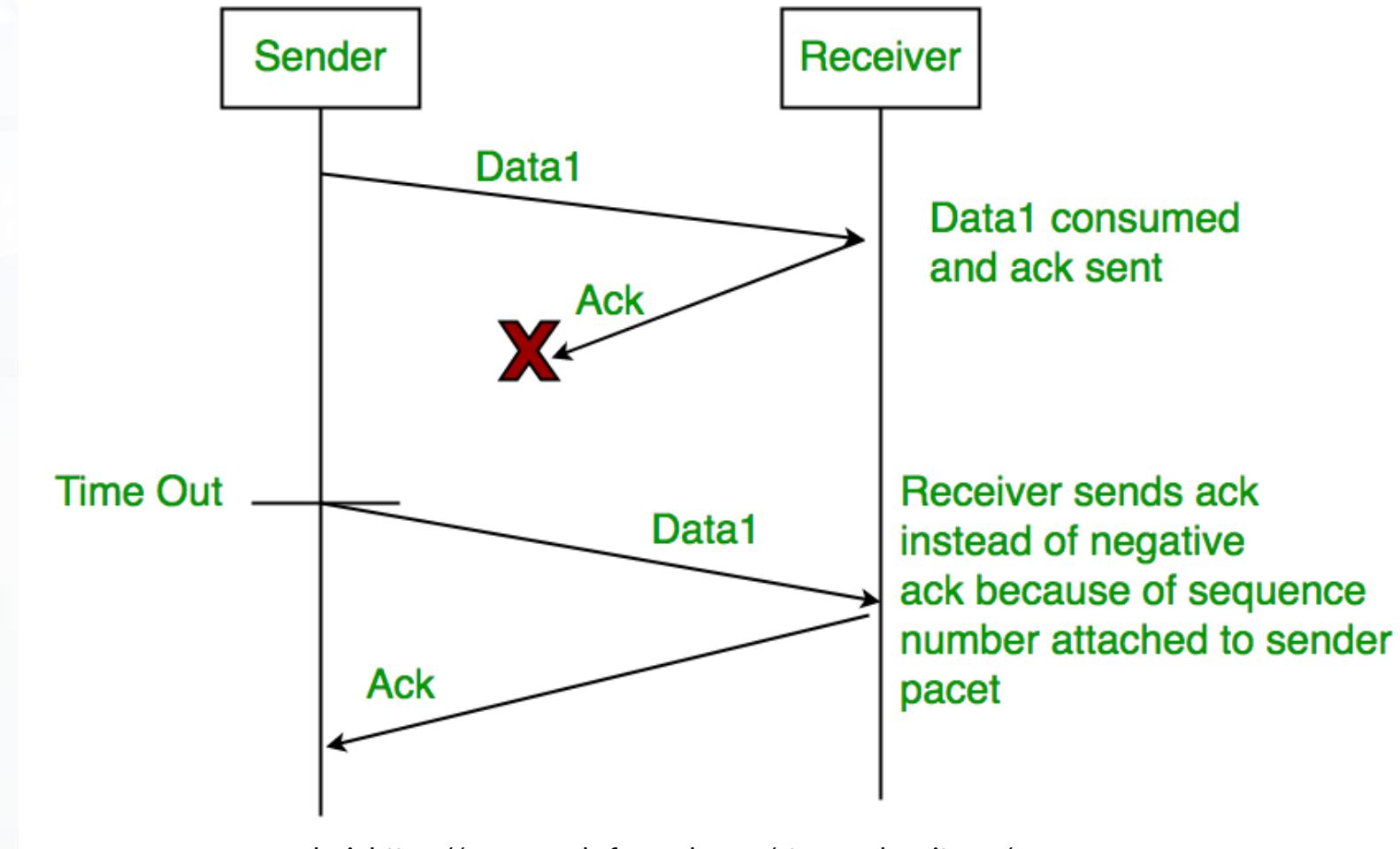
- Využívá kladné potvrzování jednotlivých rámců
- Postup přenosu
  - Odesílatel odešle data VČETNĚ zabezpečení
    - To je nutné, abych věděl jak mám rozhodnout o správnosti / chybovosti přenosu
  - Příjemce přijme data a zkонтroluje zabezpečení
    - Pokud je zabezpečení v pořádku, posílá se kladné potvrzení - ACK
    - Pokud zabezpečení není v pořádku neudělá se nic
      - Prostě data zahodíme a čekáme až přijdou znova a správné
  - Odesílatel čeká na příjem kladného potvrzení
    - Pokud jej přijme a je v pořádku začne posílat další data
    - Pokud žádné potvrzení nedorazí do určitého času - timeoutu - přenos zopakuje
    - Pokud potvrzení nedorazilo v pořádku - bylo poškozené např - odešlou se data znova a opět se čeká na potvrzení
      - Ale to je vlastně problém ... řekneme si dále

# Simplexní protokol Stop & Wait: příklad bezchybného přenosu



zdroj: <https://www.geeksforgeeks.org/stop-and-wait-arq/>

# Simplexní protokol Stop & Wait: příklad chybného přenosu

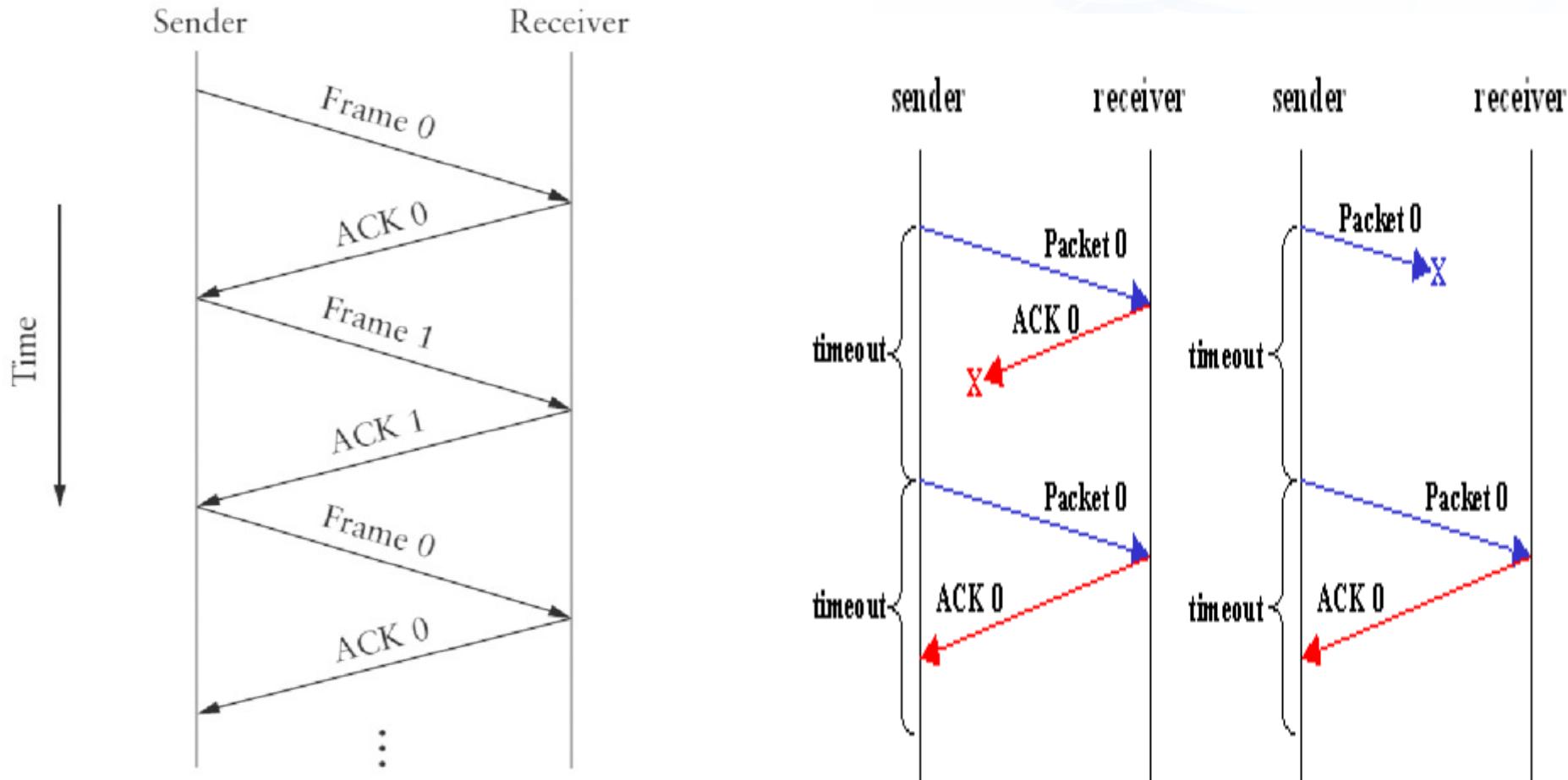


zdroj: <https://www.geeksforgeeks.org/stop-and-wait-arq/>

# Simplexní protokol Stop & Wait: číslování rámčů

- U Stop & Wait protokolu může nastat problém v případě, že
  - ACK nepřijde správně
    - Je to informační rámec a ten při přenosu poškodil
  - ACK bylo odesláno, ale nedorazí
    - Rámec vůbec nedorazí
- V obou případech odesílatel NEDOSTAL ACK v pořádku a bude přenos opakovat a odešle podruhé **stejná** data
  - To je ok a to od něj i čekáme
- Příjemce data obdrží, ale neobdrží další data jak očekával, ale dostane **znovu stejná** data
  - Na straně příjemce nastává duplicita, kterou příjemce není schopen odhalit
    - Zas by to musela řešit vyšší vrstva – opět časově drahé
- Vyřešíme zavedením číslování rámčů a potvrzení
  - Pro Frame0 přijde potvrzení ACK0, pro Frame1 bude ACK1
  - U simplexního Stop & Wait protokolu nám stačí rozlišit dva rámce Frame0 a Frame1
    - Tedy zda už posílám nový a nebo znova předchozí rámec

# Simplexní protokol Stop & Wait: číslování rámců - příklad



zdroj: [https://www.researchgate.net/figure/5-Timeline-for-Stop-and-Wait-Protocol-with-1-bit-Sequence-Number-31\\_fig18\\_236259629](https://www.researchgate.net/figure/5-Timeline-for-Stop-and-Wait-Protocol-with-1-bit-Sequence-Number-31_fig18_236259629)

zdroj: [https://www.isi.edu/nsnam/DIRECTED\\_RESEARCH/DR\\_HYUNAH/D-Research/stop-n-wait.html](https://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_HYUNAH/D-Research/stop-n-wait.html)

# Protokol Ask and Go

- Zatím jsme vždy pracovali s předpokladem, že příjemce je stále připraven přijímat data
- To ale nemusí být pravda a může být zdrojem problémů
  - Například proto, že aktuálně příjemce nemá ke zpracování dostatek paměti / dostává dat moc
- To řeší protokol Ask and Go a nebo také někdy Task and GO
- Princip je jednoduchý, napřed se zeptám zda příjemce data může přjmout a až řekne, že ano, začínám přenos
- Evidentně budu posílat nějaká data navíc
  - Dotaz a potvrzení dotazu
- Ale data „navíc“ budou výrazně menší než požadovaná data a tedy snižuji nutnost opakování přenosu
  - Což může být velice výhodné při pomalém přenosu a dlouhých blocích dat
  - Z důvodu úspory energie například v bezdrátových sítích

# Prokotol Ask and Go příklad

## Vysílač

Ptám, se zda mohu vysílat

## Přijímač

ASK1 →

← ACK1

Odpovídám, že ano a  
očekávám data

Odesílám data

DATA1 →

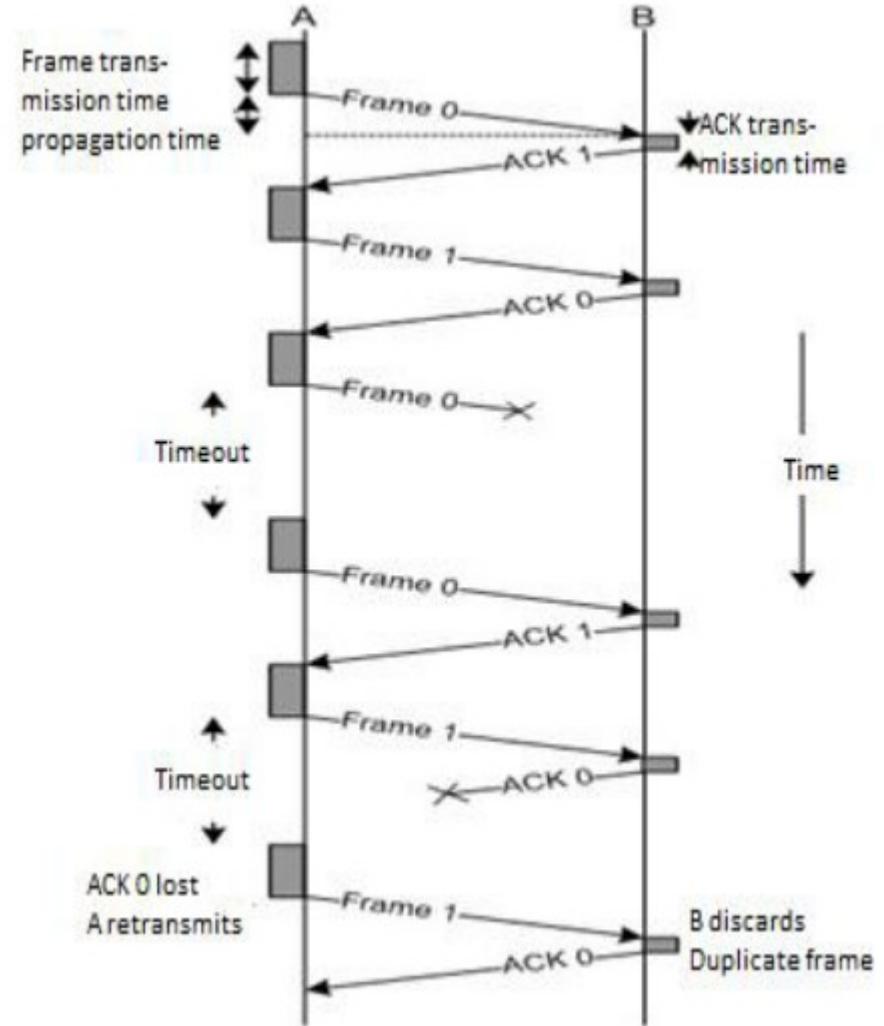
← ACK1

Data jsou v pořádku a  
posílám kladné hodnocení

Data jsou přijata v pořádku  
mohu pokračovat s dalším rámcem

# Jednotlivé a kontinuální potvrzování

- Protokol Stop & Wait používá jednotlivé potvrzování
  - Každý jeden rámec je odeslán, pak je čekáno na jeho potvrzení a AŽ po kladném potvrzení se pokračuje dále
  - Jednoduché na implementaci, na zdroje – málo paměti
  - Používán byl například v IPX/SPX společnosti Novell
  - Jedená se o polo-duplexní komunikaci
  - Vede k velice špatnému využití přenosového kanálu, zvlášť při vyšším RT
    - $n=t_{\text{TRANS}}/(t_{\text{PROP}}+t_{\text{TRANS}}+t_{\text{PROP}}+t_{\text{ACK}}) [\%]$ 
      - Pro LAN ( ethernet ) kde je latence linky ~25.6 μs je efektivita ~ 92%
        - » Což je ok
      - Pro WAN kde je latence linky např 50ms je efektivita ~ 2.3%
        - » Což je velice špatně
- Řešením je „kontinuální“ potvrzování
  - Nečekám po každém odeslání na potvrzení, ale posílám ihned další data
  - Potvrzení přichází se zpožděním

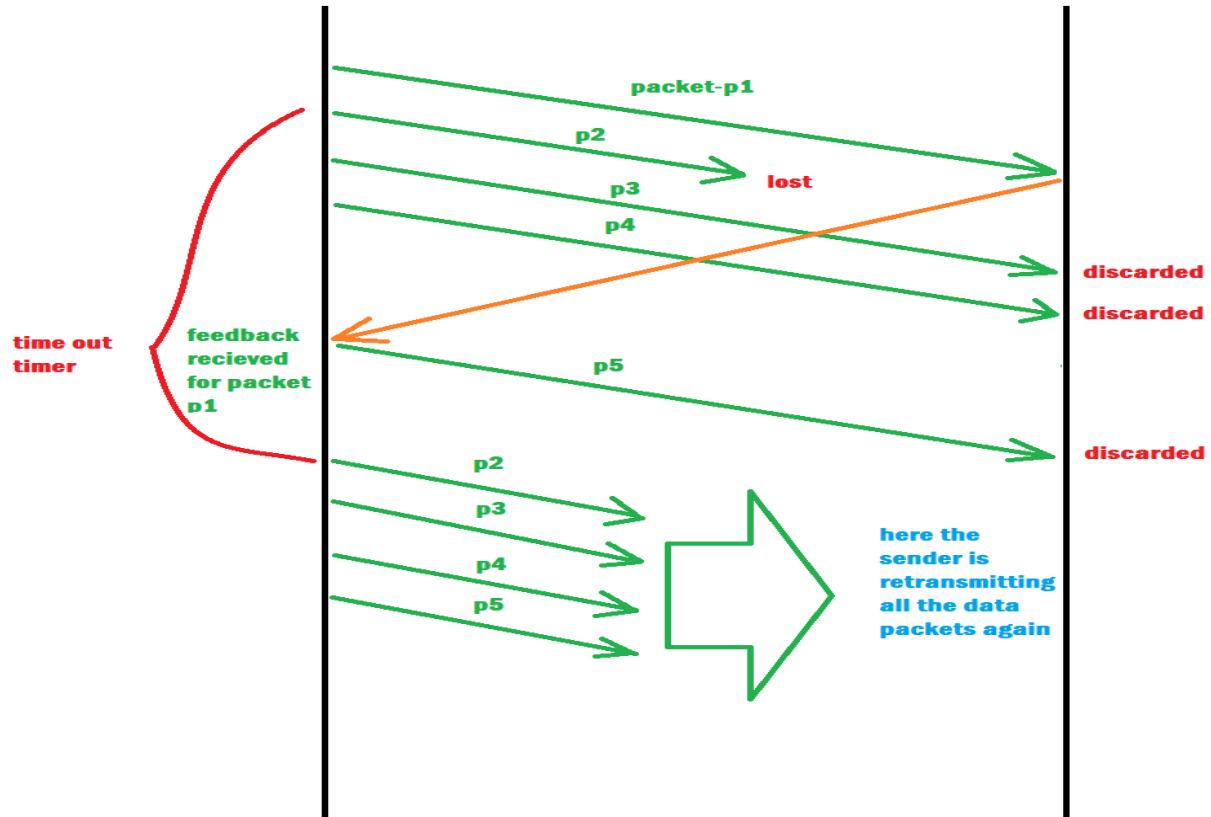


zdroj: [https://www.researchgate.net/figure/Link-utilization-and-working-process-of-stop-and-wait-ARQ\\_fig1\\_275043653](https://www.researchgate.net/figure/Link-utilization-and-working-process-of-stop-and-wait-ARQ_fig1_275043653)

# Kontinuální potvrzování: Go Back N

- Nečekání na přijetí ACK1 pro Frame1, ale posílá další data ihned po odeslání Frame1
- Lepší využití linky, protože nedochází k čekání bez přenosu
- To, že data i potvrzení mohou chodit na „přeskáčku“ umožňuje duplexní přenos
- Vyžaduje číslování rámců a potvrzení
  - Rámce mohou být posílány v různém pořadí – v případě opakování přenosu – a tedy je potřeba je identifikovat
- Princip fungování
  - Pokud se příjmou data ve správném pořadí a mají v pořádku zabezpečení, pošle se kladné potvrzení a data jsou postoupna vyšší vrstvě na další zpracování
  - Pokud se příjmou data se správným zabezpečením, ale MIMO pořadí ( čekám Frame2, ale přišel Frame3 ), tato a následující data jsou zahozena
  - Jelikož příjemce neobdržel Frame2 a tedy neposlal ACK2, odesilatel pošle Frame2 po vypršení timeoutu znova a zároveň i všechny následující rámce
- Velmi často se v komunikaci nevyskytuje jediná chyba osamoceně, ale bývá poškozeno více rámců zároveň – na což Go Back N reaguje
- Špatné využití přenosového kanálu, protože při chybě 1 rámce se jich odesílá znova N, přestože jejich přenos už mohl být úspěšný

# Kontinuální potvrzování: Go Back N - příklad

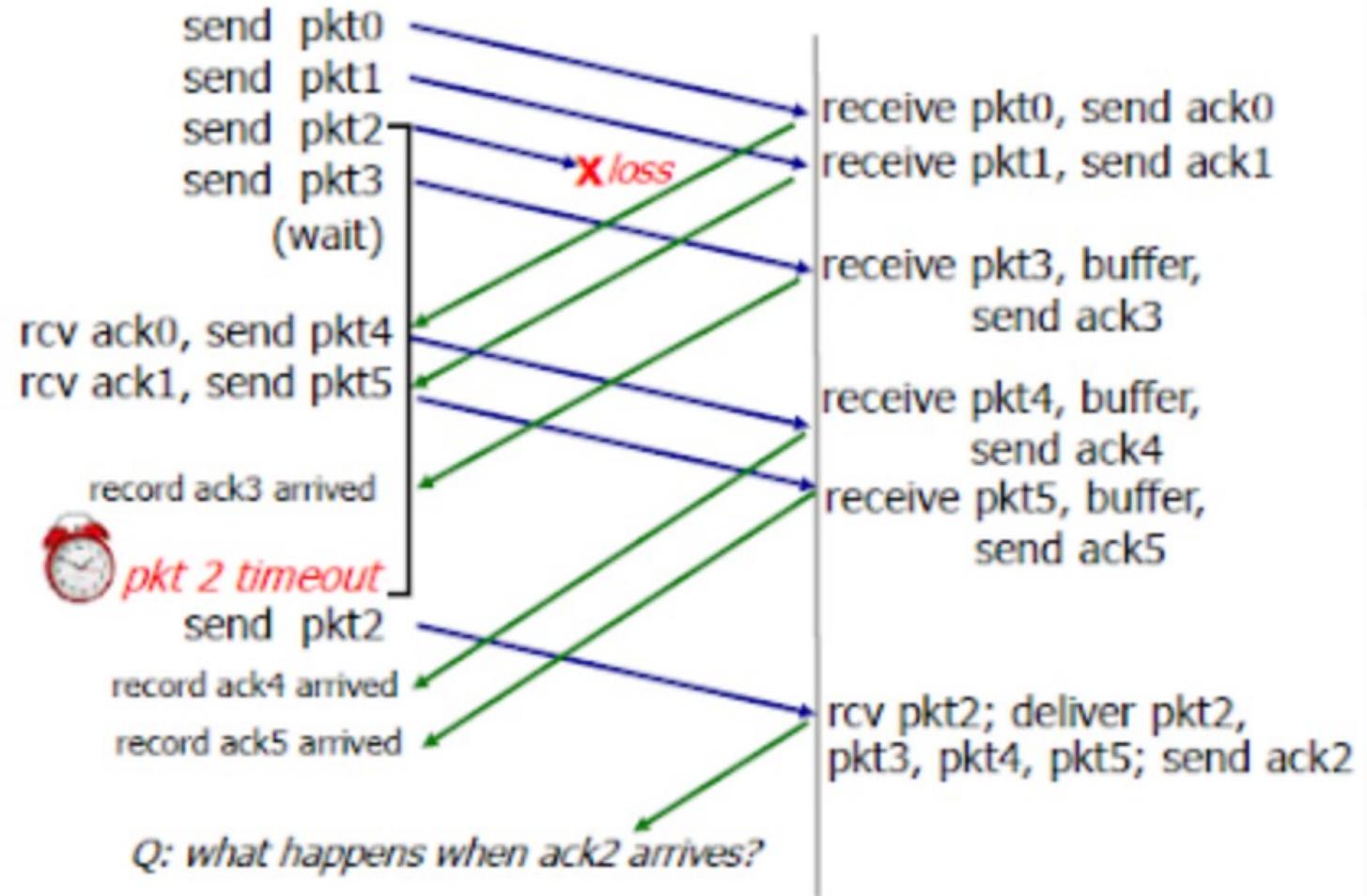


zdroj: <https://www.studytonight.com/post/flow-control-gobackn-arq-protocol>

# Kontinuální potvrzování: Selective repeat

- Jak bylo uvedeno Go Back N lépe využívá kapacitu přenosového kanálu než Stop&Wait, ale v případě chyby přenáší veškerá data od chyby znova – což může být neefektivní
  - Ale nemusí, záleží na charakteru linky, tedy jaký typ chyb převažuje, zda samostatné a nebo sdružené
- Řešením je kontinuální selektivní kladné potvrzování, které stejně jako Go Bank N nečeká na potvrzení pro právě odeslaný rámec, ale posílá ihned další data
- Na rozdíl od Go Back N, ale při chybě posílá znova JEN chybný rámec
  - Nedochází k duplicitním přenosům a tedy se lépe využívá kapacita přenosového kanálu
  - Na straně příjemce musí existovat buffery, kde se uchovávají rámce došlé mimo pořadí, protože vyšší vrstvě L3 se musí odesílat ve správném pořadí

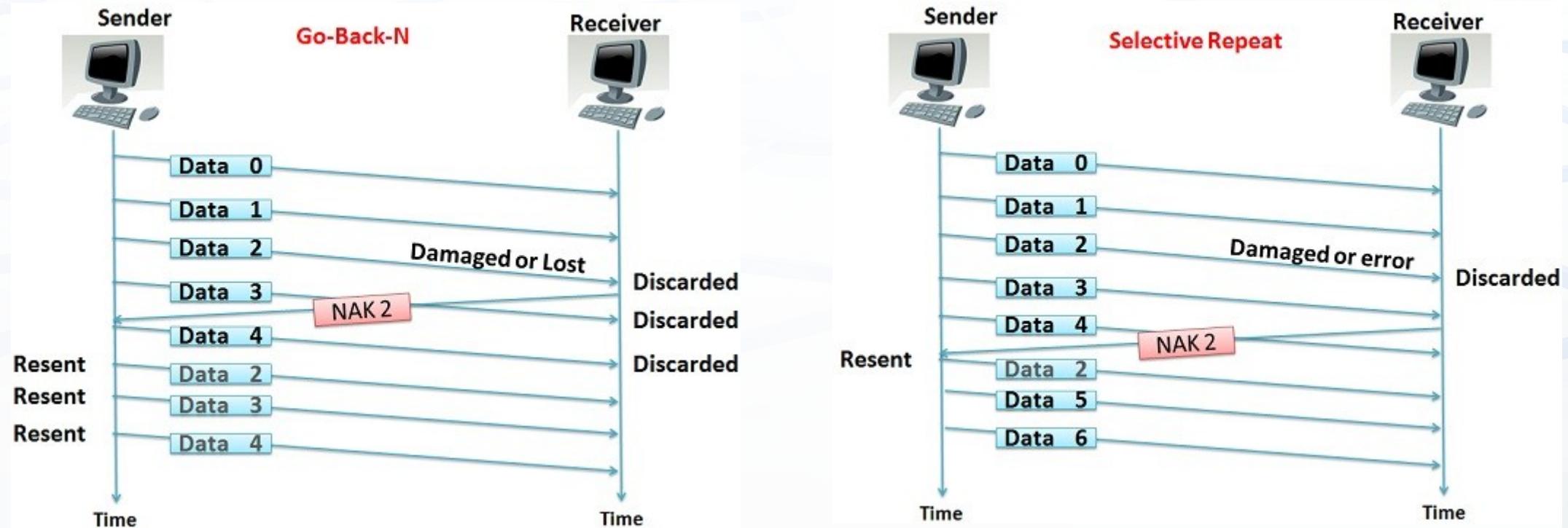
# Kontinuální potvrzování: Selective repeat - příklad



# Kladné a záporné potvrzování

- Jak jsme v předchozích případech uváděli, tak ACK – kladné potvrzení slouží k potvrzení správného přijetí dat
- Pokud kladné potvrzení nepřijde včas – tedy do uplynutí timeoutu – je přenos považován za chybný a dle protokolu je opakován
  - Někde jen jeden rámec( Stop&Wait ) někde i více rámců(Go Back N )
- Pokud data nedorazila správně, neděje se při kladném potvrzení nic a čeká se na timeout
- To ale může být problém, protože timeout musí být vyšší než je doba nutné k odeslání dat a přijetí potvrzení
  - Jako příjemce musím dlouho čekat než odesílateli dojde, že nastala chyba o které já už vím
- Řešením je kombinace kladného a záporného potvrzování, protože pak se i informace o chybě přenáší ihned jak se existence chyby zjistí
- Typicky se záporné potvrzování označuje jako NACK nebo někdy NAK

# Kladné a záporné potvrzování - příklad



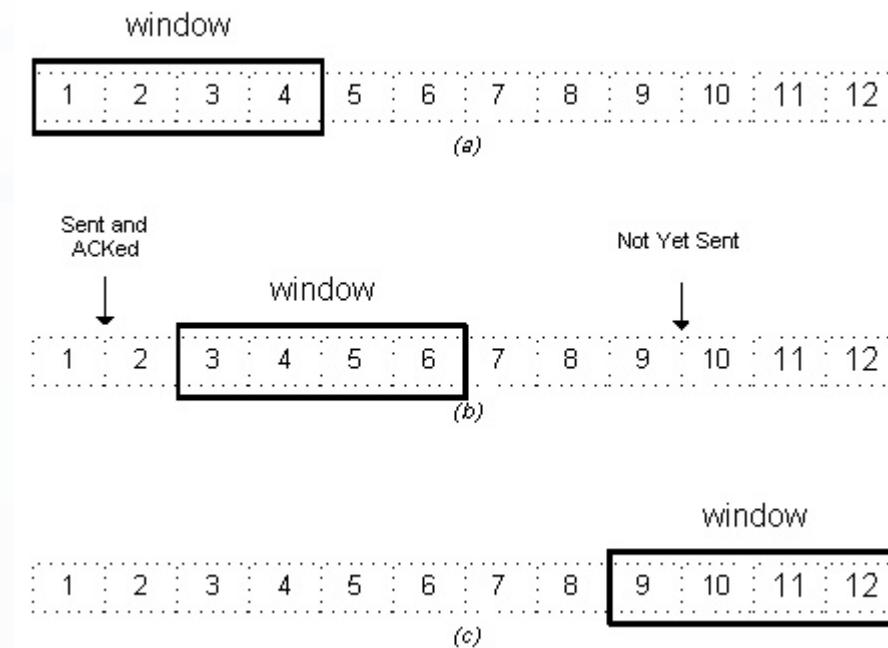
zdroj: <https://techdifferences.com/difference-between-go-back-n-and-selective-repeat-protocol.html>

# Protokoly s klouzajícím okénkem

- Při použití kontinuálního potvrzování může odesíatel využít maximálně kapacitu přenosového kanálu – což ale ne vždy musí být žádoucí
  - Příjemce nemusí stačit data zpracovávat
    - Z důvodu výkonu
    - Z důvodu nedostatečné paměti pro uchování rámců ve frontě na zpracování
- Pokud příjemce data nestačí zpracovávat nemá ( podle toho co zatím víme ) jinou možnost než nadbytečná data zahodit
  - Například proto, že je nemá kam uložit
- Do hry opět vstupuje timeout
  - Kterého už jsme se snažili kombinací kladného a záporného potvrzování zbavit a který nám komunikaci výrazně zpomaluje
- Základní myšlenka je, že poslat data znova je často nákladnější, než je poslat jednou i když pomaleji než je kapacita linky, ale tak, že je příjemce stačí zpracovat

# Protokoly s klouzajícím okénkem II.

- Nebudeme odesílat libovolné množství dat – počet rámců – ale jen „pevně“ definovaný počet
- Tento počet – blok – nám definuje „okénko“ tedy „kolik dat najednou vidíme“
- Odesílám rámce ihned za sebou jen do velikosti okénka
  - Tím, že nám limituje pohled na data tak jich více „nevidíme“
- Po přijetí ACK1 posuneme okénko o 1 pozici vpravo
- Rychlosť posunu okénka se mění podle příchod ACK
- Pokud nepřijde ACK1 nemůžeme okénko posunout, protože může dojít k opakovanému přenosu
- Příjemce nemusí mít neomezeně buffery, protože dle velikosti okénka ví kolik dat MAX najednou může dostat

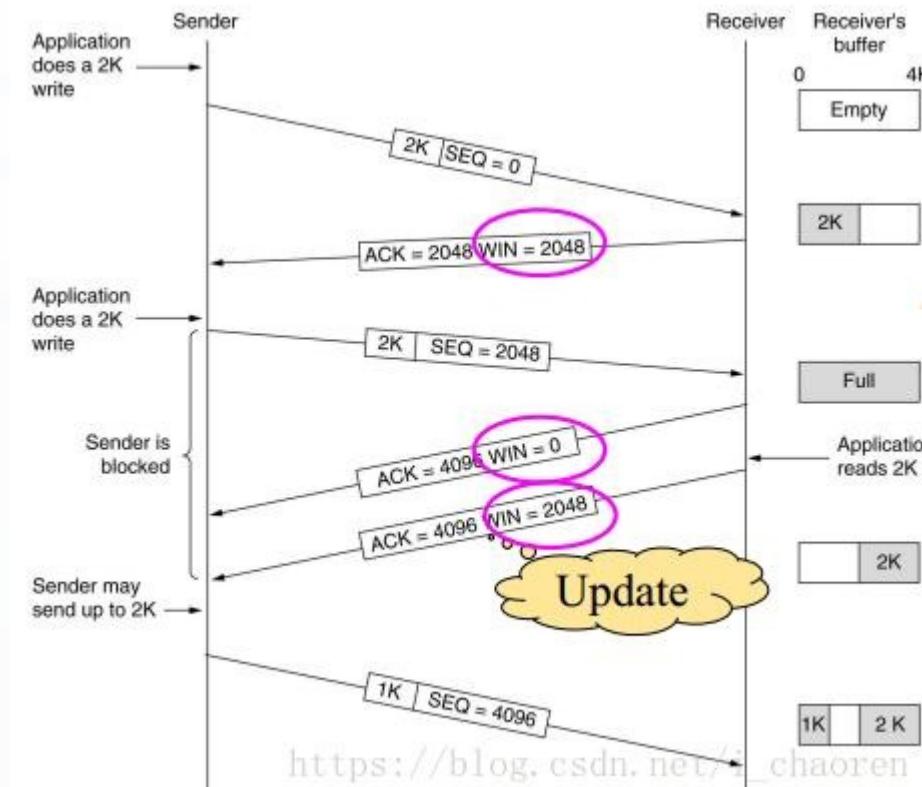


zdroj:

<https://www.thecrazyprogrammer.com/2017/05/sliding-window-protocol-program-c.html>

# Protokoly s klouzajícím okénkem: řízení rychlosti II.

- Příjemce se spolupodílí na stanovení velikosti
  - Odesílatel jej nemusí respektovat / může navrhnut vlastní hodnotu
- Je možné realizovat jako samostatné rámce a nebo jako součást datových rámců
- Řízení rychlosti se může řešit na více vrstvách
  - Například L4 - TCP
- Řízení rychlosti řeší dva problémy
  - Zahlcení příjemce
    - Kapacita sítě je dostatečné, ale příjemce nestíhá
      - Odesílatel a příjemce mohou být HW výrazně rozdílní
      - Příjemce může být využíván více klienty
  - Zahlcení sítě
    - Cesta mezi odesílatelem a příjemce je saturovaná nebo chybí
    - Data nedorazí vůbec nebo dorazí chybně



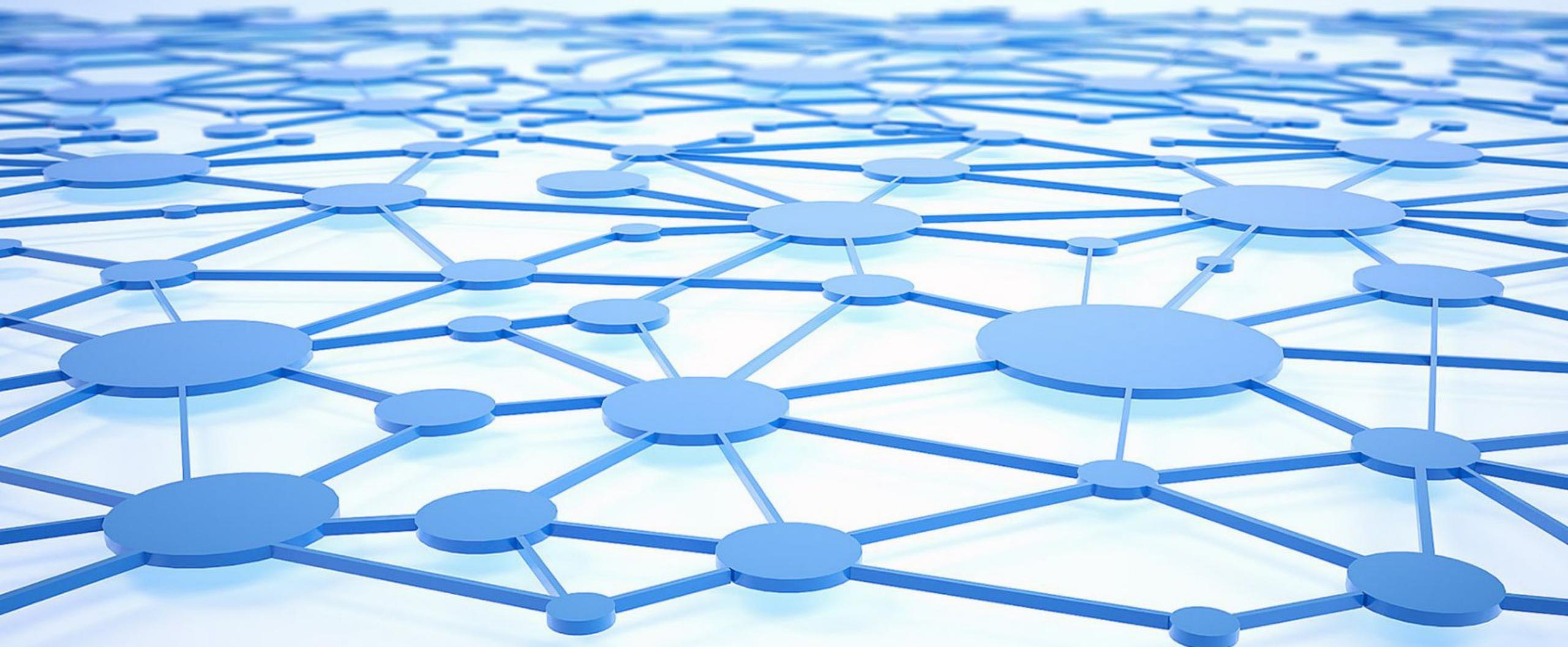
# Protokoly s klouzajícím okénkem: řízení rychlosti III – reakce na zahlcení

- Zahlcení příjemce
  - Pokud je úplné nemůže příjemce dělat nic a data zahodí
    - Odesílatel nedostane ACK a ví, že musí snížit rychlosť
  - Pokud má příjemce ještě prostor reagovat, pošle WIN s nižší hodnotou než je aktuální
    - Tím dokáže rychlosť snížit
    - Může poslat i WIN0 a tím dočasně přenos pozastavit
    - K obnovení přenosu dojde na základě zprávy od příjemce s WIN > 0
- Zahlcení sítě
  - Příjemce odeslal NACK
    - Data dorazila, ale jsou chybná
    - Odesílatel pošle data znovu A ZÁROVĚN sníží WIN, protože očekává problém v síti
    - Toto se může opakovat
  - Odesílateli nedorazilo ani ACK ani NACK nebo došla poškozená
    - Data se ztrácejí nebo přenos chybí
    - Stejně jako v předchozím případě odesílatel reaguje snížením rychlosť, protože očekává že při stejně rychlosći se bude s větší pravděpodobností chyba opakovat

# Úvod do počítačových sítí

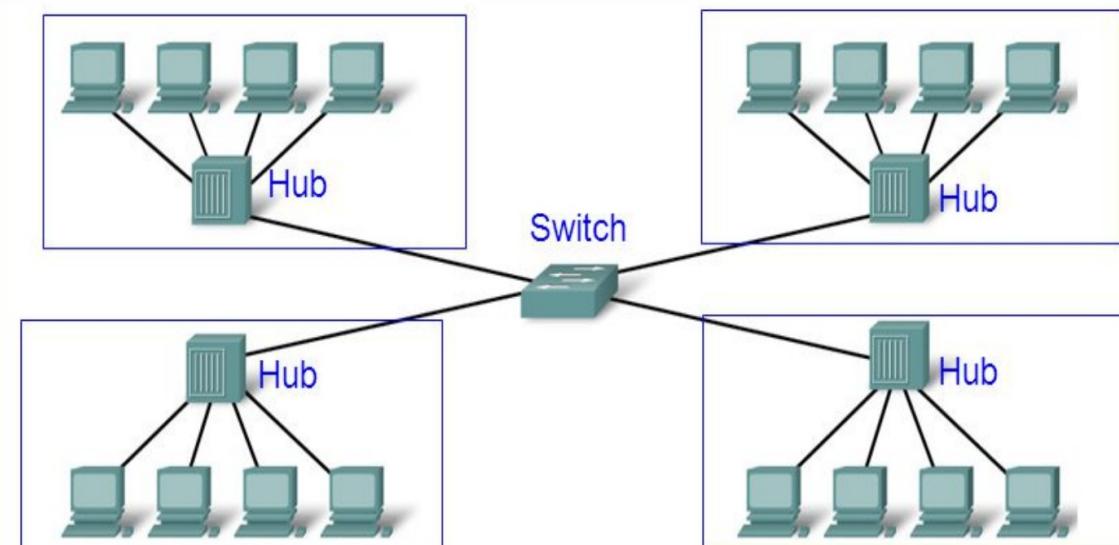
Přednáška 7 ( 2025/2026 )

ver. 2025-10-21-01



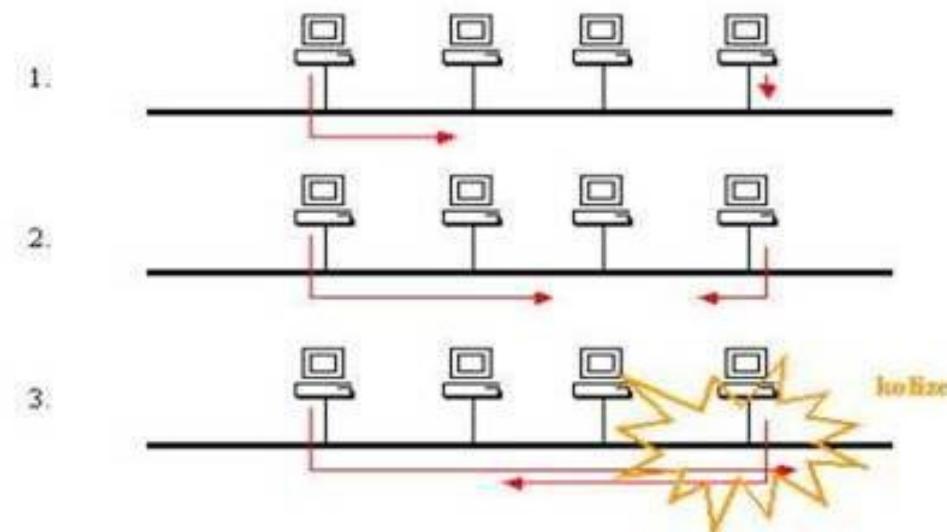
# Řízení přístupu

- MAC – Media Access Control
- Snažíme se vyřešit situaci, kdy N zařízení chce používat sdílený komunikační kanál
  - Bezdrátové sítě P-MP – Point to Multi-point
    - LAN: WiFi, Bluetooth
    - WAN: WiMAX, mobilní síť
  - Drátové sítě – bez ohledu na topologii či vodič
    - LAN: Ethernet, Token ring
    - WAN: DOCSIS
- Definujeme kolizní doménu – počet stanic, které mohou vzájemně kolidovat / oblast kde ke kolizi může dojít
  - HUB – všechny porty
    - Hub == více-portový opakovač
    - Vysílaná data vidí všichni, ale ne nutně ve stejný čas
  - Switch – vždy jen jeden port
    - Kolize může teoreticky nastat jen na jednom portu
    - Vysílání se nešíří na všechny porty
      - V rámci broadcastu ano, ale pak se jedná o kopii rámce na jednotlivé porty, ne o kopii signálu
- Na obrázku máme 4 kolizní domény
  - Každý port switche je jedna kolizní doména
  - Všechny porty každého hubu jsou v jedné kolizní doméně



# Řízení přístupu: Kde je problém

- Problém je v „překrývání“ či „slučování“ signálů
  - Dva signály se potkají a z pohledu pozorovatele vznikne třetí - výsledný
- Pokud do společného kanálu začnou vysílat dvě stanice jejich vysílání se se vzájemně ovlivní a nebude možné je **zpětně oddělit**
- Vysílací stanice nemusí o kolizi ihned vědět
  - Signál není v jeden okamžik po celém komunikačním kanálu
    - Vliv zde hrají fyzikální vlastnosti, například rychlosť šíření světla
  - Stanice odvysílá svá data a může mít „pocit“, že vše proběhlo správně, ale to nemusí být pravda



# Řízení přístupu versus multiplex

- Vypadá to podobně – více účastníků chce současně použít jeden komunikační kanál, ALE
  - Multiplex:
    - Je na fyzické vrstvě
    - Máme dva přístupové body a N účastníků
    - Účastníci jsou na jednom místě
    - Všichni se snaží přenést data z bodu A do bodu B stejným kanálem
  - Řízení přístupu:
    - Je na linkové vrstvě
    - Máme přenosový kanál – např sběrnici – a na ní N účastníků
    - Účastníci od sebe mohou být různě vzdáleni – signál k nim nedorazí ve stejný okamžik
    - Každý účastník může chtít přenášet data k různému cíli
      - A pro B, C pro D, D pro A

# Řízení přístupu: Cíle

- Snažíme se o :
  - Předcházení kolizím
    - Předcházení problémům je téměř vždy „levnější“ než jejich následné řešení
  - Pokud už kolize nastane, potřebujeme to co nejdříve zjistit
    - Teprve když o problému víme, můžeme jej začít řešit
  - Pokud jsme zjistili, že nastala kolize je třeba zastavit vysílání a dát všem vědět o problému
    - Aby se minimalizoval čas, kdy se nepřenáší / přenáší již poškozená data
  - Snažíme se předcházet opakované kolizi
    - Pokud dvě stanice začnou vysílat v jeden okamžik dojde ke kolizi, pokud se následně vysílání přeruší – opět ve stejný okamžik - a znova začne, situace se bude **pravděpodobně** opakovat

# Typy přístupu k mediu: Nevýlučný

- Vysílací stanice nemá k mediu výlučný přístup – není sama kdo může vysílat
- Komunikační kanál sice sdílí s dalšími, ale má k němu trvalý přístup
- Metody nevýlučného přístupu jsou velice podobné metodám používaným v multiplexu či se dá říci, že z nich vycházejí
- Je možné garantovat přenosovou rychlosť – logicky protože vím, kdy a jak budu přenášet
- Jde o princip přepojování okruhu, kdy „sestavím“ cestu a tou posílám data – proud bitů
  - Nejedná se z tohoto **pohledu** o strukturovaná data – rámce / pakety
- Příklady:
  - FDM ( Frequency Division Multiplexing )                          => FDMA ( Frequency Division Multiple Access )
  - TDM( Time Division Multiplexing )                          => TDMA ( Time Division Multiple Access )
  - CDM( Code Division Multiplexing )                          => CDMA( Code Division Multiple Access )

# Typy přístupu k mediu: Nevýlučný - příklady

- FDMA ( Frequency Division Multiple Access )
  - Dělí pásmu na jednotlivé kanály dle frekvencí - např po 22 MHz u WiFi
  - Zde je nevýlučný přístup - v rámci spektra
  - V rámci jednotlivých kanálů je ale přístup také třeba řešit -
    - CDMA / TDMA
- TDMA ( Time Division Multiple Access )
  - Vytváří time sloty v rámci jednoho FDMA kanálu
- CDMA( Code Division Multiple Access )
  - Vysílá na stejném kanále, ale oddělitelnými „ortogonálními chipping“ kódy
    - Jak jsme si už říkali u CDM
- Frequency Hopping
  - Vysílání velice rychle mění frekvenční kanály
  - Změna je pseudonáhodná a je daná pro každou dvojici odesilatel/příjemce => šance, že dojde ke kolizi je velice malá
    - Pokud už k ní dojde, je opět třeba ji řešit, zde například až na vyšší vrstvě
  - Používá například Bluetooth

# Typy přístupu k mediu: Výlučný

- Vysílací stanice **má** k mediu výlučný přístup – vysílám sám
  - Respektive by jej chtěl mít
- Mám k dispozici celý komunikační kanál, ale jen na omezenou dobu
  - Je třeba se „střídat“
- Jde o princip přepojování „paketů“ ( zde samozřejmě rámců )
  - Odesílám každý jeden ucelený blok strukturovaným dat a pak se řeší co dále
- S tímto typem přístupu není v ISO/OSI počítáno – proto se L2 „dělí“ na dvě
  - MAC a LLC
- Výlučné přístupy můžeme dělit dle:
  - „Předvídatelností“
    - Deterministické a nedeterministické
  - Dle způsobu řízení
    - Centralizované a distribuované
  - Ostatní
    - Dotazovací, předávací, rezervační, soutěžní a další

# Typy přístupu k mediu: Výlučný přístup: Dělení dle převídatelnosti

- Deterministický – bez prvků náhody – předvídatelný
  - Vždy vím přesně co se bude dít
  - Řídí se pravidly, která vždy vedou v konečném čase k cíli
  - Stejná posloupnost událostí skončí vždy stejným, výsledkem
  - Je garantováno, že každé stanice, která požaduje vysílat to bude v konečném čase umožněno
  - Nevýhodou je dražší implementace a často i „dražší“ provoz z pohledu využití sítě, pokud se jedná o řídký provoz
  - Příkladem je např Token ring
- Nedeterministický – s prvků náhody – nepředvídatelný
  - Do řešení vstupuje náhoda – například v podobě náhodného odkladu dalšího vysílání
  - Není garantováno, že každé stanici, která požaduje vysílat, to bude umožněno
    - S „téměř“ sto procentní jistotou ano, ale na sto procento říct nemůžeme - „blbá shoda náhod“
  - Stejná posloupnost událostí může dopadnout různě
    - Ale to může být právě výhoda – například při čekání na opakování vysílání
  - Příkladem je např Ethernet

# Řízení přístupu: Výlučný: Dělené dle způsobu řízení: Centralizovaný

- Máme nějakého arbitra / moderátora - obecně stanici, která řídí provoz
- Jednoduché na realizaci
- Jednoduchá možnost změny přidělování práva vysílat - adaptivní vlastnosti
- Možnost zohledňovat priority
  - Jeden bod rozhoduje o tom kdo teď smí vysílat a některé stanice upřednostňovat - dávat jim více prostoru
- Problém při výpadku moderátora
  - Při jeho výpadku celá síť přestává fungovat
  - Je třeba detekovat výpadek a provést volbu/náhradu moderátora
- Může se stát, že moderátorů je více
  - To je chyba a je třeba mít možnost tuto situaci detekovat a reagovat na ni
- Dokáže garantovat přístup pro stanici v konečném čase
  - Protože se typicky chová deterministicky, ale nutně nemusí
- Příkladem je např síť 100 VG Any-LAN

# Řízení přístupu: Výlučný: Dělené dle způsobu řízení: Distribuovaný

- V síti nemáme žádnou řídící stanici, ale používáme společný algoritmus
- Zcela eliminujeme problém výpadku či volby moderátora – není třeba
- Všechny stanice musí algoritmus dodržovat, jinak to nemůže fungovat
  - Nedodržení může nastat například vinou chyby HW
  - Je třeba dodržování pravidel kontrolovat a na případné nedodržení reagovat
- Těžko se mění pravidla fungování sítě, protože je třeba novou informaci předat všem
  - Složité je to proto, že se často jedná o HW implementaci
- Může se chovat deterministicky i nedeterministicky
  - Ale jen jedním zvoleným způsobem v daném stavu
- Příkladem je Ethernet nebo Token ring

# Řízení přístupu: Výlučný přístup: Další možnosti

- Dotazovací
  - Typicky, ale ne nutně, u centralizovaných řešení
    - Centrální stanice se ptá zda někdo nechce vysílat
    - Stanice, které chce vysílat se ptá centrální stanice nebo všech ostatních zda může vysílat
- Předávací
  - Jednotlivé stanice si „předávají pověření“ - právo vysílat - často označované jako „pešek“
  - Pokud mám právo vysílat mohu tak učinit, ale po pevně dané době musím „peška“ předat zas dále
    - Vzniká problém při ztrátě „peška“
- Rezervační
  - Sítí nejprve koluje „rezervační“ rámec, do kterého stanice provedou záznam, pokud budou chtít vysílat
  - Až rámec obejde celou síť, začnou vysílání dle rezervací
  - Opět problém při ztrátě rezervačního rámec
- Soutěžní
  - Stanice společně soupeří o přístup k médiu dle pevně daných pravidel

# Řízení přístupu: Výlučný přístup: Příklady: „Čistá“ Aloha

- Aloha je metoda řízení přístupu ke sdílenému mediu, která vznikla na Havajských ostrovech kolem roku 1970
  - Cílem bylo realizovat radiové spojení mezi jednotlivými ostrovy vzdálenými až 600km
- Princip byl velice jednoduchý
  - Kdo potřebuje vysílat tak to prostě udělá bez ohledu na cokoliv dalšího
  - Jelikož se na další neohlížím, může dojít ke kolizi – souběžnému a neoddělitelnému vysílání
    - Pokud se tak stane, operace musí být zrušeny, protože překrývající se vysílání nejdou oddělit
    - Nepoznám zda došlo ke kolizi nebo chybě – každopádně dle kontroly nejsou přenesená data správně a tedy neposílám potvrzení
  - Pokud ke kolizi nedošlo – tedy přenos proběhl správně – posílám kladné potvrzení
  - Pokud potvrzení nedojde v rámci timeoutu, přenos opakuji
    - Je jedno zda kvůli kolize či jiné chybě, prostě vysílám znovu – předtím ale náhodnou dobu počkám – abych riziko opakované kolize snížil
    - Pokud to udělá více stanic naráz, problém se bude velice pravděpodobně opakovat
- Reálně fungovalo jen díky řídkému provozu
  - I dnes se ještě používá tam, kde je velké zpoždění přenosu ( vysoké RTT ) - například satelitní přenos
- Díky velice častým kolizím má velice nízké využití kapacity přenosového kanálu – cca 18%
  - Logicky, čím více kolizí, tím častější potřeba přenos opakovat nebo čekat před začátkem dalšího přenosu a tím více „promrhaného“ času

# Řízení přístupu: Výlučný přístup: Příklady: „Slotted“ Aloha

- Slotted Aloha je také někdy označována jako synchronní Aloha
- Jak bylo řečeno, Aloha má velice špatné využití kapacity přenosové kanálu, což pochopitelně s nárůstem komunikace nevyhovovalo
- Důvodem bylo velké kolizní okénko
  - Tedy čas kdy může dojít k chybě
  - Rovnalo se dvojnásobku velikosti vysílaných dat
  - To bylo dáno tím, že stanice mohly vysílat úplně kdykoliv
- Řešením bylo sjednotit začátek vysílání – tedy minimalizovat čas souběhu
  - Centrální stanice vysílá všem info, kdy se může začít vysílat
  - Vysílají se pevně dané bloky – velikostně/časově
  - Ke kolizi tedy nedojde vůbec a pokud ano, tak se budou téměř 1 k 1 překrývat celé bloky
    - A tím bude velikost kolizního oken rovna MAX velikosti / délce vysílaných dat – tedy 1x lepší než u čisté Aloha
- Využití kapacity přenosového kanálu je zde až 36 % - tedy 1x více než u čisté Aloha

# Řízení přístupu: Výlučný přístup: Příklady: CSMA

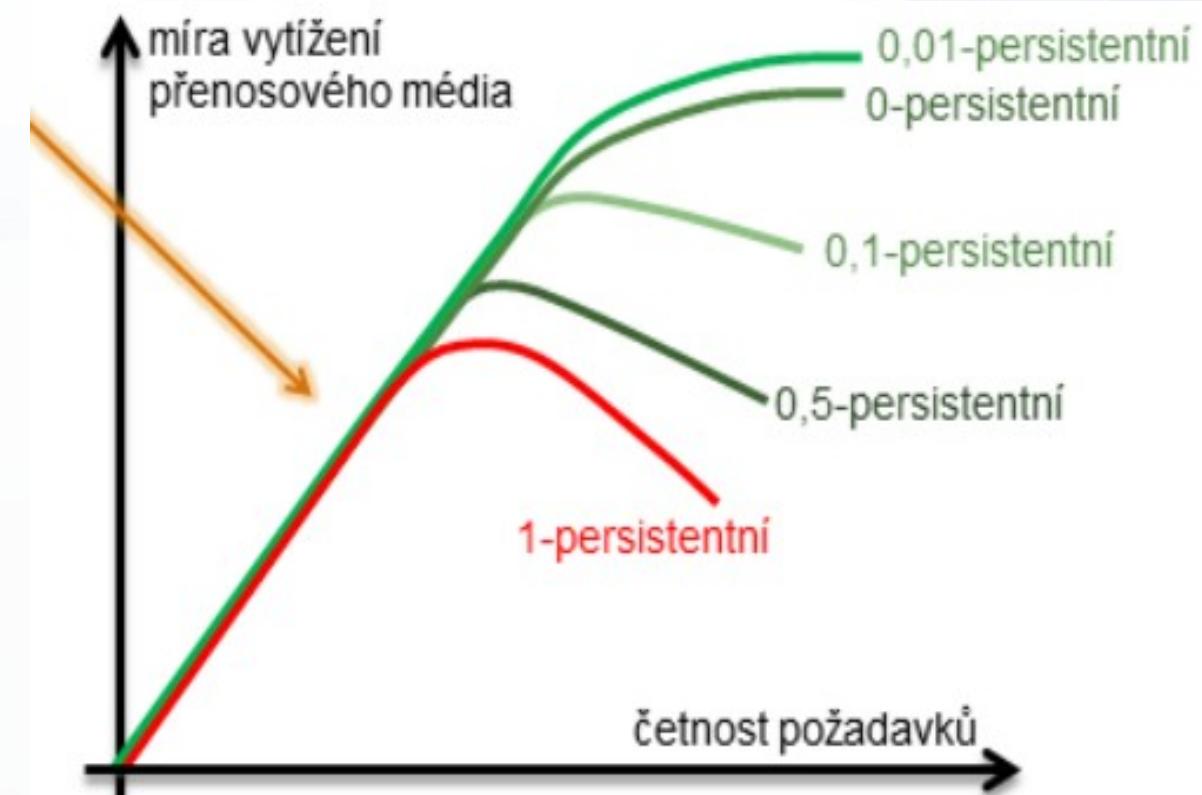
- CSMA - Carrier Sence Multiple Access
- Skupina přístupových metod, které se snaží minimalizovat možnost vzniku kolize „nasloucháním nosné“
  - Než začnu vysílat zkontroluji přenosový kanál, zda už náhodou neprobíhá vysílání
    - Pokud ne, mohu začít vysílat já
    - Pokud ano, vysílání musím odložit a zkusit po čase kontrolu znova
- Samozřejmě zde může dojít ke kolizi
  - Dva se „rozhlédneme“ ve stejný čas a pokud je vše v pořádku začneme vysílat
  - Pokud ke kolizi dojde, zjistí se to až po dokončení celého přenosu a kontrole zabezpečení

# Řízení přístupu: Výlučný přístup: Příklady: CSMA - naléhavost

- Pokud zjistím, že nějaký přenos probíhá počkám na jeho konec a pak pokračují podle jednoho ze tří možných scénářů
  - **Naléhající / vytrvalý / persistentní**
    - Nutně potřebuji vysílat, takže čekám na konec běžícího přenosu a jak je dokončen začnu sám vysílat
    - Zde už neprobíhá znova ověření, takže pokud se stejně chová více stanic, téměř jistě dojde ke kolizi
    - Výhodou je, že se „neztrácí“ čas a tím může docházet k lepšímu využití kapacity přenosového kanálu / nízká latence
    - Použito například u CSMA/CD v rámci Ethernetu
  - **Nenaléhající / „nevytrvalý“ / ne-persistentní**
    - Pokud poslechem nosné zjistím, že vysílat nemohu, vzdám to rovnou a na náhodně dlouhý čas se odmlčím
    - Pokud to udělají i další stanice, a čas čekání je náhodný, výrazně eliminuji pravděpodobnost kolize
    - Ale zhorší se využití kapacity přenosového kanálu( zvýší se latence ), protože jistě vzniknou místa, kdy budou všichni náhodně dlouho čekat, ale linka už bude volná
    - Použito například i CSMA/CA v rámci WiFi
  - **P-Naléhající / „hodit si kostkou“ / p-persistentní**
    - Kombinace dvou předchozích – s pravděpodobnosti X se zachovám persistentně
    - Cílem je zvýšit využití kapacity přenosového kanálu, ale zároveň příliš nezvyšovat pravděpodobnost kolize okamžitým přenosem
    - Tedy ano, ke kolize může dojít, ale ne tak často a využití kanálu se zvýší, byť ne na maximum
      - Tedy kompromis

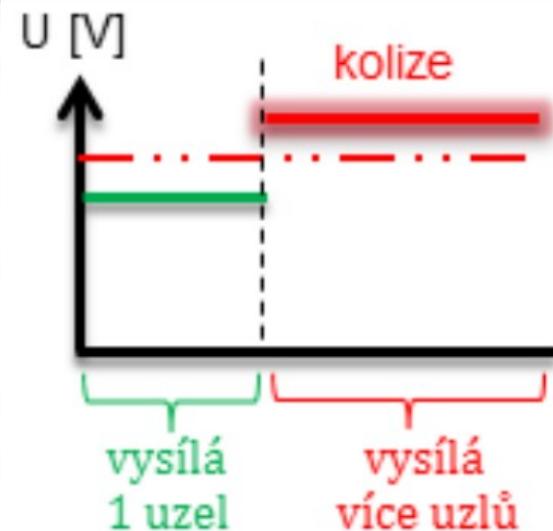
# Řízení přístupu: Výlučný přístup: Příklady: CSMA – naléhavost - praxe

- Potřebujeme ideálně najít kompromis v naléhavosti
- Lze předpokládat, že čím více požadavků na vysílání a kolizí, tím hůře
  - Bude třeba více času k opakovaní přenosů
- Nejhůře se tedy budou chovat naléhající varianta
- Nejlépe pak vychází p-naléhající, kde p je rovna 0,01
- Při nízkých počet požadavků na přenos se všechny varianty chovají stejně
  - Protože je požadavků málo, je pravděpodobné, že budou rozprostřené v čase, kolize nebudou vnikat přirozeně

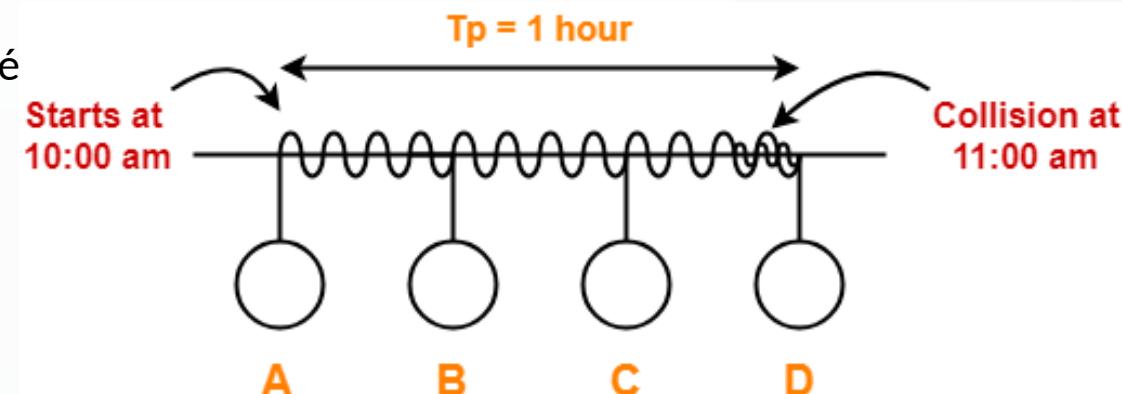


# Řízení přístupu: Výlučný přístup: Příklady: CSMA/CD

- Nevýhodou CSMA je fakt, že o kolizi je odhalena až po dokončení vysílání
  - Dvě stanice ověří příposlechem nosné, že linka je volná a začnou vysílat
  - Obě - či více - začnou vysílat v téměř stejný okamžik a pokračují dokud neodvysílají celý blok
  - Po odvysílání je zkontrolováno zabezpečení rámce, zde je nalezena chyba a řeší se oprava
  - Je to zdlouhavé a nákladné na čas – kapacita využití přenosové kanálu klesá
- Řešením je snaha detekovat kolizi už v průběhu přenosu a tím minimalizovat čas násobného přenosu
  - Pokud dojde ke kolizi, nemá cenu pokračovat, stejně už to skončí s chybou, je třeba co nejdříve začít znova - pokud možno lé
  - CD – Collision Detect – společně s odesláním dat kontroluje, zda nepřichází data od jiné stanice
    - Signál, který se spojí z více stanice bude „nestandardní“  
- tedy takový, který by se na lince neměl vyskytovat
    - Pokud ano, došlo ke kolizi



zdroj: <https://www.eearchiv.cz/l226>



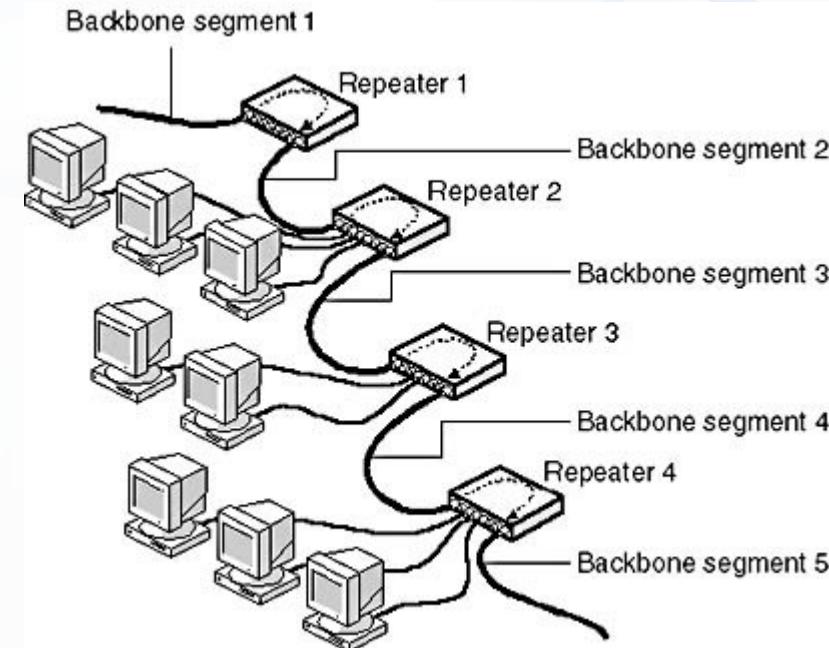
zdroj: <https://www.gatevidyalay.com/csma-cd-access-control-in-networking/>

# Řízení přístupu: Výlučný přístup: Příklady: CSMA/CD : JAM

- Pokud v rámci CSMA/CD stanice zjistí, že došlo ke kolizi postupuje následovně:
  - Ukončí své vysílání
    - Proč by také pokračovala, už ví, že přenos dobře neskončí a tak může minimalizovat čas počátku obnovy
  - Rozešle do sítě JAM signál
    - Jednotlivé stanice nepřijmou data ve stejný okamžik
      - Kvůli vzdálenosti od vysílače a fyzikálním vlastnostem šíření signálu
    - Stanice, která pošle JAM tak pomáhá ostatním v co nejkratším čase detekovat problém
  - S opakovaným vysíláním počká náhodně dlouhou dobu
    - Princip náhodné „rozstřelu“ – aby nové vysílání nezačalo opakovaně u více stanic ve stejný čas
      - Samozřejmě při vyšším počtu stanic se to i tak může stát, ale pravděpodobnost je nižší
  - Pokud i opakovaný pokus, po náhodně dlouhém čekání, selže, čekám znova, ale tentokrát už dvakrát tak dlouhý čas
    - První čekání nepomohlo, tak raději počkám déle, abych zvýšil šanci, že už linka bude volná
    - Samozřejmě může dojít ke snížení využití kapacity přenosového kanálu, ale to je menší zlo než nutnost opakovaného vysílání N stanic

# Řízení přístupu: Výlučný přístup: Příklady: CSMA/CD : JAM : délka

- Pokud má JAM signál správně zafungovat, je třeba aby se s jistotou dostal „včas“ ke všem stanicím v rámci jedné kolizní domény
  - Pokud by byl kratší, došlo by k tomu, že část stanic by JAM neviděla mohla začít vysílat
- JAM signál je vlastně speciální rámec, který musí být minimálně tak dlouhý, aby v jeden okamžik obsáhl celou kolizní doménu
  - Tím pádem musí být omezena velikost kolizní domény
    - Nemohu ji pomocí opakovačů rozširovat neomezeně, protože tím by musel být delší JAM a tím by docházelo v případě kolize k větším latencím na lince
    - Uvádí se pravidlo 5:4:3
      - 5 segmentů – tedy spojů mezi opakovači
      - 4 opakovače – tedy maximálně 4 za sebou zapojené opakovače signálu
      - 3 sítě / obydlené segmenty – bloky mezi opakovači, kde je stanice
    - Např pro 10Mbps Ethernet je to 512bitů



zdroj: <https://flylib.com/books/en/2.733.1.31/1/>

# Řízení přístupu: Výlučný přístup: Příklady: CSMA/CD : příklad Ethernet

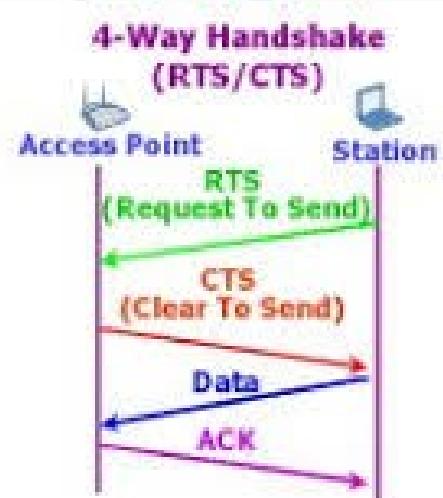
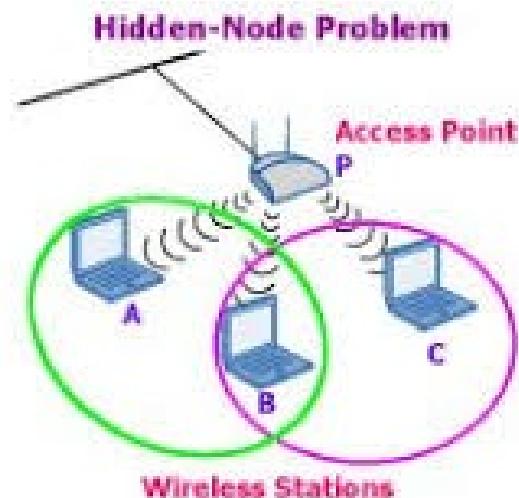
- Ethernet ve variantě half-duplex využívá CSMA/CD
  - Pro koaxiální kabel například
  - Pro full-duplex už není třeba, protože velikost kolizní domény je 1
- Rámcem nesmí být kratší než 64B
  - Pokud by byl, docházelo by falešné identifikaci volné linky
- Používá se JAM o délce právě 64B, tedy 512 bitů
- Maximální přenosové rychlosti pro koaxiální kabel – do 10Mbps
  - Zde je vidět neefektivita tohoto řešení, protože obecně lze koaxiální kabel použít na výrazně vyšší rychlosti – viz přenos pomocí rozvodů kabelové televize

# Řízení přístupu: Výlučný přístup: Příklady: CSMA/CA

- CSMA/CA - Carrier Sense Multiple Access with Collision Avoidance
  - Collision Avoidance – vyhýbání se kolizím
- Používá se tam kde není možné použít CD – Collision Detect
  - Například bezdrátové sítě
- Princip chování se může dle implementací lišit, ale vždy se snaží kolizi předejít, protože ji nemůže v průběhu přenosu rozpoznat
- Příklad pro LocalTalk
  - Provedu kontrolu nosné a poté pošlu info ostatním stanicím, že chci vysílat
  - Žádost o vysílání je typicky násobně menší, než přenášená data, takže mohou nastat dvě situace:
    - Žádost projde v pořadu => Fajn, všichni ví, že já chci vysílat a tedy tedy by ke kolizi nemělo dojít
      - Samozřejmě ještě zde může být problém nové stanice, ale ta šance je výrazně nižší
    - Žádost neprojde v pořádku a je třeba ji opakovat => Ok, to je problém, ale ztratil jsem násobně méně času, než při vysílání reálných dat
- Příklad pro WiFi( IEEE 802.11 )
  - Kontroluji, zda je linka volná
  - Pokud ano, počkám náhodně dlouhou dobu, pak zkontroluji zda je stále volná
    - Pokud ano, zkusím odeslat data a čekám na potvrzení doručení
  - Pokud ne, čekám dvojnásobnou dobu a pokus opakuji
  - Pokud odvysílám data a nepřijde včas potvrzení, opět čekám náhodně dlouhou dobu na opakování přenosu a pokud vysílat nemohou, zdvojnásobím čas čekání a zkusím znovu

# Řízení přístupu: Výlučný přístup: Příklady: CSMA/CA s RTS/CTS

- Samotné čekání nemusí zajistit, že ke kolizi nedojde
- U CSMA/CA v rámci bezdrátových sítí se jako nepovinný prvek může ještě použít RTS/CTS
  - Vše předchozí, tedy příposlech nosné, čekání po zjištění volné linky atd zůstává platné
- Chceme ještě více snížit riziko kolize, takže provedeme „rezervaci“ kanálu
  - RTS – Request To Send – Dotaz na vysílání
    - Podobně jako u LocalTalk, než začnu vysílat, optám se, zda je mohu
    - Stanice, která chce vysílat, se zeptá přístupového bodu ( například AP ) zda může
  - CTS – Clear To Send – Volno k odeslání
    - Po přijetí odpovědi může začít vysílat, protože ví, že nikdo jiný právo vysílat v daný čas nemá
- Kromě jiného řeší problém skrytého uzlu
  - Typický problém v bezdrátových sítích, kdy se ne všechny uzly musí vidět navzájem
  - Snadno pak dojde k situaci, kdy stanici detekovala volný kanál, ale ono to tak nemusí být pro další stanice



zdroj: [https://www.marigold.cz/wifi/doku.php/problem\\_skryteho\\_uzlu?  
rev=1154167313](https://www.marigold.cz/wifi/doku.php/problem_skryteho_uzlu?rev=1154167313)

# Řízení přístupu: Výlučný přístup: Příklady: CSMA/BA

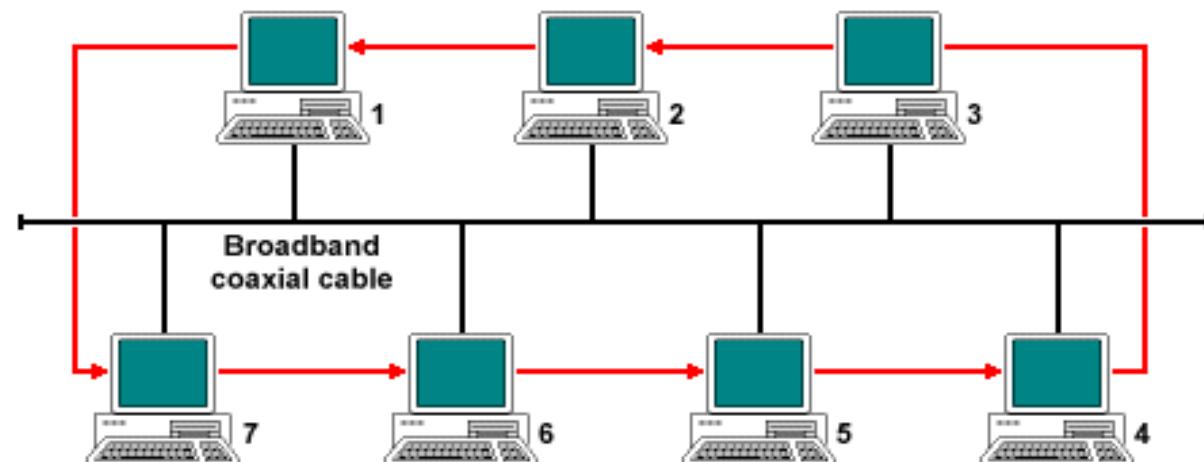
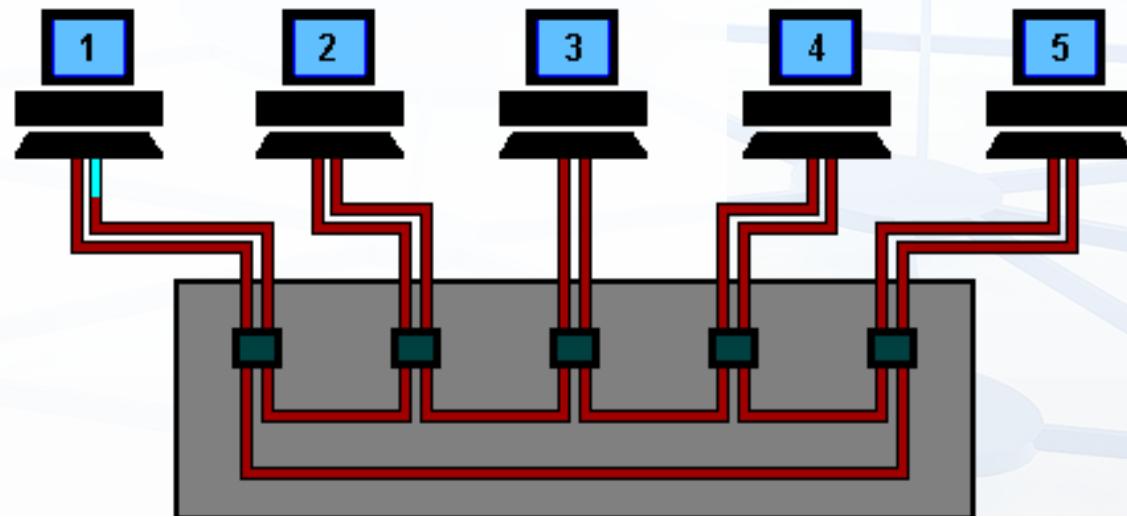
- Carrier Sense Multiple Access with Bitwise Arbitration
  - Bitwise Arbitration - bitová arbitráž / řešení priorit
- Opět řešíme příposlech nosné, ale v případě kolize nečekání náhodnou dobu jako v CSMA/CD
- Jednotlivé stanice mají přiřazen kód – prioritu
- Pokud dojde ke kolizi, je na základě „priority“ vybrána stanice, která smí vysílat
- Používání v průmyslu – CAN sítě
  - Například v autech na komunikaci řídících jednotek
- Výhodou je nižší prodleva po kolizi
- Navíc je možné upřednostňovat určité typy zpráv / rámců
  - Například info o nutnosti zapnout ABS má jistě přednost před info o prasklé žárovce ...
- Vzniká zde ale problém monopolizace
  - Pokud bude docházet ke kolizím často a u stejných stanic, nemusí se ty s nižší prioritou k vysílání vůbec dostat
  - Řešením je priorita složená z více prvků nebo proměnlivá v čase

# Metody řízení přístupu: Výlučný přístup: Token Passing

- Principem je předávání pověření / „peška“
  - Kdo má „peška“ může vysílat
- Jedná se deterministickou distribuovanou metodu
- Aby předávání fungovalo, je nutné aby bylo zapojení kruhové
  - Fyzicky – opravdu do kruhu zapojená síť
  - Logicky – zapojení je jiné než kruhové a „nějak jinak“ je zajištěn kruh
    - Pomocí MUA při zapojení do hvězdy
    - Pomocí pořadí ve sběrnicových sítích
      - Zde je třeba řešit výpadek stanice – protože pokud jedna vypadne, síť přestane fungovat
      - Stejně tak příchod nové stanice
- Problém nastává v případě, že
  - Pověření se ztratí
  - Pověřených koluje více
- Obě situace řeší jeden vyčleněný prvek – aktivní monitor
  - Ten je typicky volen například na základě nejvyšší MAC
  - Při výpadku aktivního monitoru je zas nutné zvolit jiný

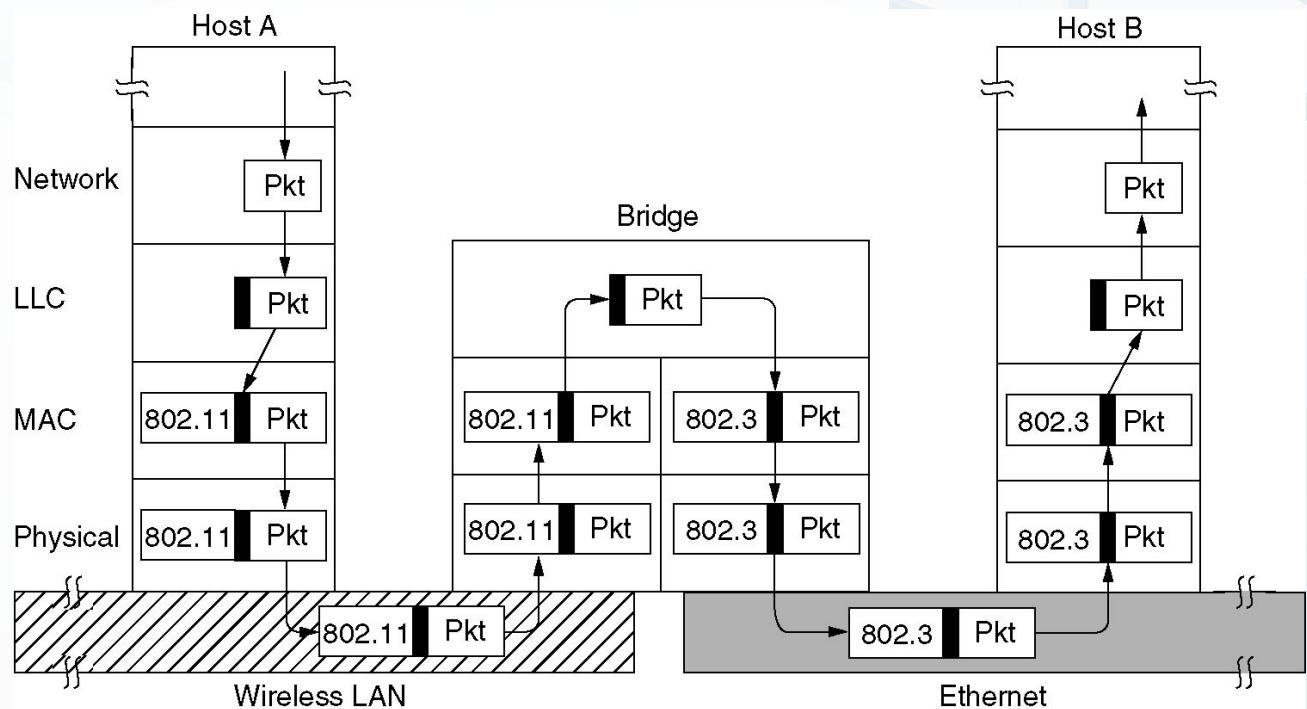
# Metody řízení přístupu: Výlučný přístup: Token Passing : příklady

- Token Ring
  - Dvě varianty
    - IBM Token ring – zapojení do hvězdy, kroucená dvojlinka
    - IEEE 802.5 – nepředepisuje žádnou topologii ani medium
  - Používá logický kruh
  - Při větší síti a zatížení je efektivnější než Ethernet
  - Pokud nikdo nevysílá, koluje prázdné pověření
- Token Bus
  - Využívá sběrnicovou topologii
  - Kruh je opět pouze logický

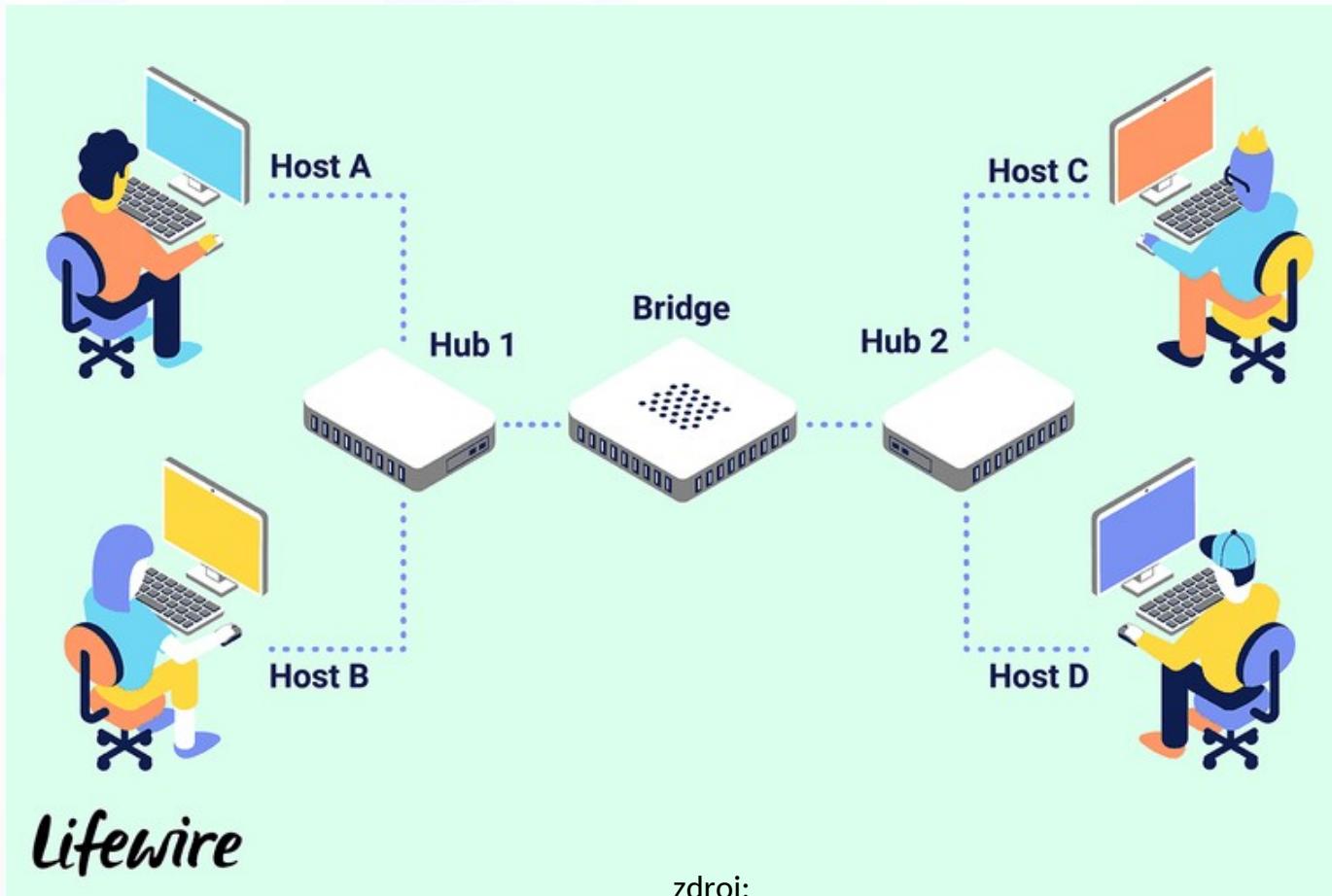


# Mosty mezi 802.x a 802.y

- V rámci L2 se vyskytuje více protokolů
  - Mají své rámce, svůj přístup k mediu, své ověření ...
- Jejich vzájemnou komunikaci zajišťují mosty / bridge mezi protokoly
  - Jedná se vlastně o zařízení, které umí oba protokoly
  - Dochází pak k transformaci jednoho rámce do rámce jiného
  - Například přechod Ethernet  $\leftarrow \rightarrow$  WiFi
    - Zde je i jiné přenosové medium atd.
- Mosty mohou být:
  - Lokální
    - Propojení v rámci LAN
      - Např dva segmenty 802.3
  - Vzdálené
    - Propojení více LAN prostřednictvím WAN
      - Např Ethernet over PPP
- Existují dvě možné konfigurace
  - Transparentní – samoučící se
    - Na začátku nezná o síti nic, ale postupně se učí
    - Pokud je zdroj i cíl ze stejné sítě nedělá nic
    - Pokud neví, pošle data dále
    - Zmenšuje kolizní doménu, ale nezmenšuje broadcastovou doménu
  - Se zdrojovým směrováním
    - Použití například v Token ring sítích
    - Kromě cílové adresy musí být uvedeno i to, přes jaké všechny mosty mají data projít



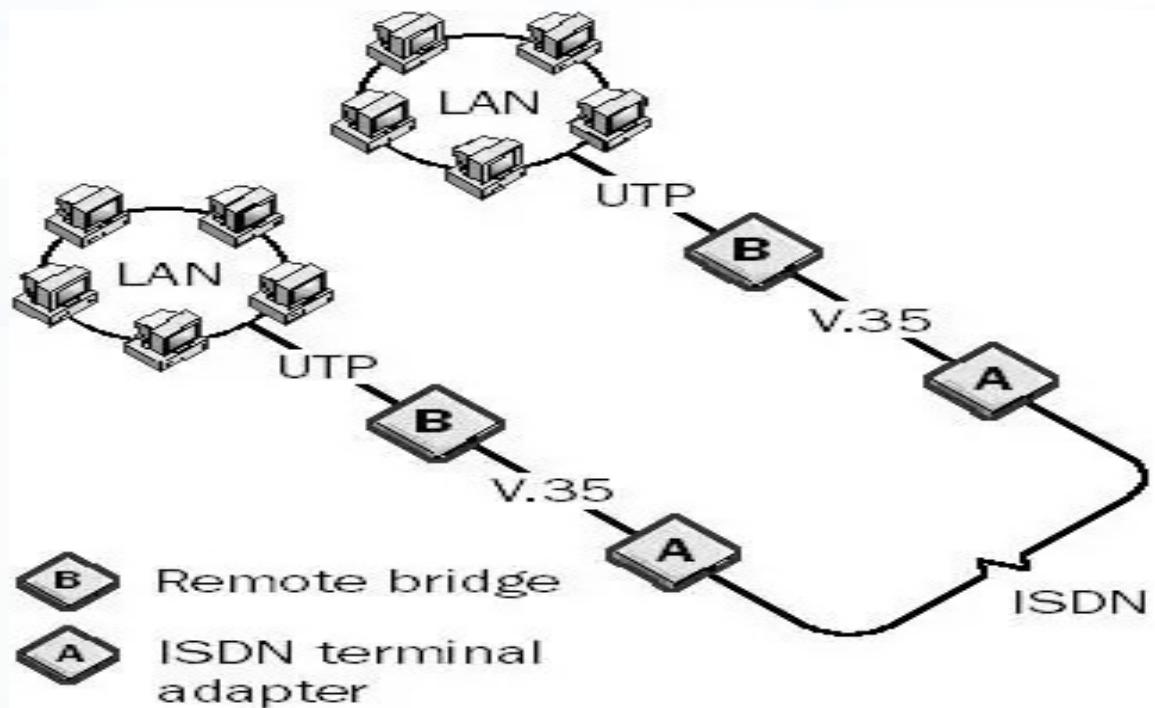
# Mosty mezi 802.x a 802.y: příklad lokálních mostů



zdroj:

[https://signup.fishedfun.com/en/html/sf/registration/eone\\_m3dsc.html#&sf=eone&lng=en&m=books&ref=5261516&prod=2&sub\\_id=explain-network-bridge-diagram&\\_sign=1a3ae649ec918155aa3976c2785a09ac&\\_signt=1606300992&utm\\_ex](https://signup.fishedfun.com/en/html/sf/registration/eone_m3dsc.html#&sf=eone&lng=en&m=books&ref=5261516&prod=2&sub_id=explain-network-bridge-diagram&_sign=1a3ae649ec918155aa3976c2785a09ac&_signt=1606300992&utm_ex)

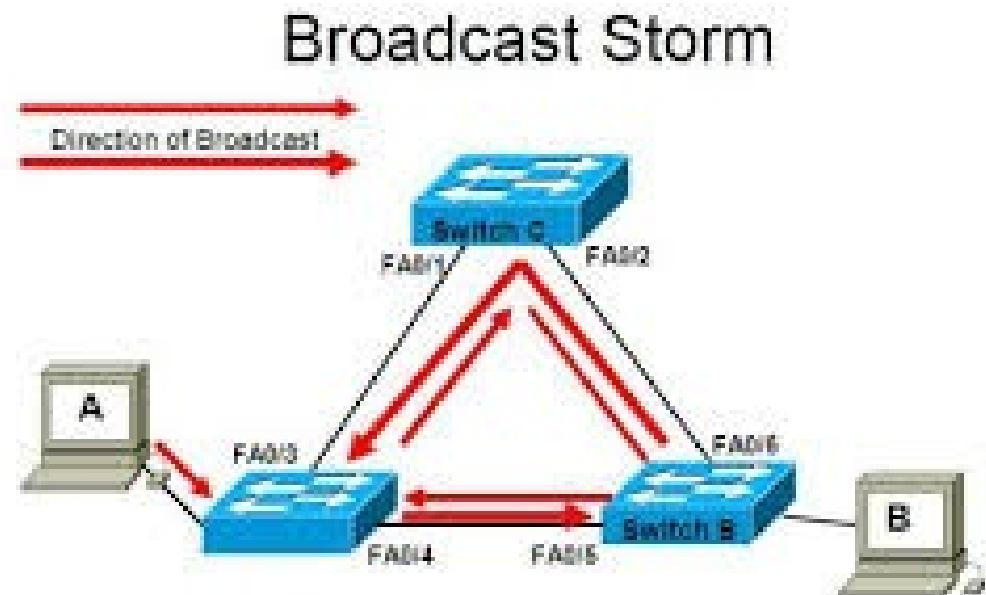
# Mosty mezi 802.x a 802.y: příklad vzdálených mostů



zdroj: [https://networkencyclopedia.com/remote-bridge/?utm\\_content=cmp-true](https://networkencyclopedia.com/remote-bridge/?utm_content=cmp-true)

# Spanning tree protokol

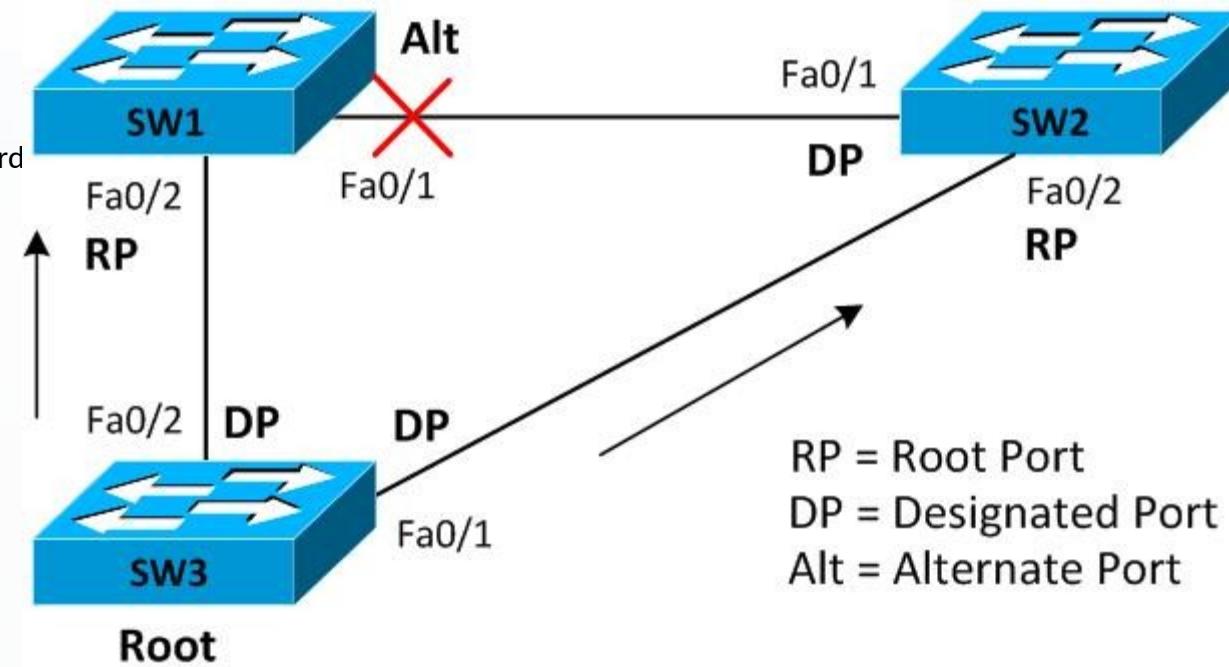
- STP – Spanning tree protocol
- Pokud propojíme více typu HUB / Bridge / Switch a nebo jen více portů jednoho zařízení, může dojít k zacyklení
- Typický problém u broadcastových vysílání
  - Ty jsou na všechny porty kromě příchozího
  - Vznikají broadcastové bouře
    - Přetížení zařízení díky násobnému doručování zpráv
- Cílem STP je hledat smyčky v síti a ty SW rozpojujeme
  - Samozřejmě pokud to zařízení umí
- 
- Problém ve velkých sítích – konverze velice dlouho trvá
- Kromě prevence smyček může sloužit i jako fail-over řešení
  - Násobně zapojené linky jsou SW odpojené do výpadku aktivně používané
- Příklady:
  - Multiple Spanning Tree Protocol
  - Rapid Spanning Tree Protocol
  - Shortest Path Bridging



- Host A sends a broadcast.
- Switches continue to propagate broadcast traffic over and over

# Spanning tree protokol - fungování

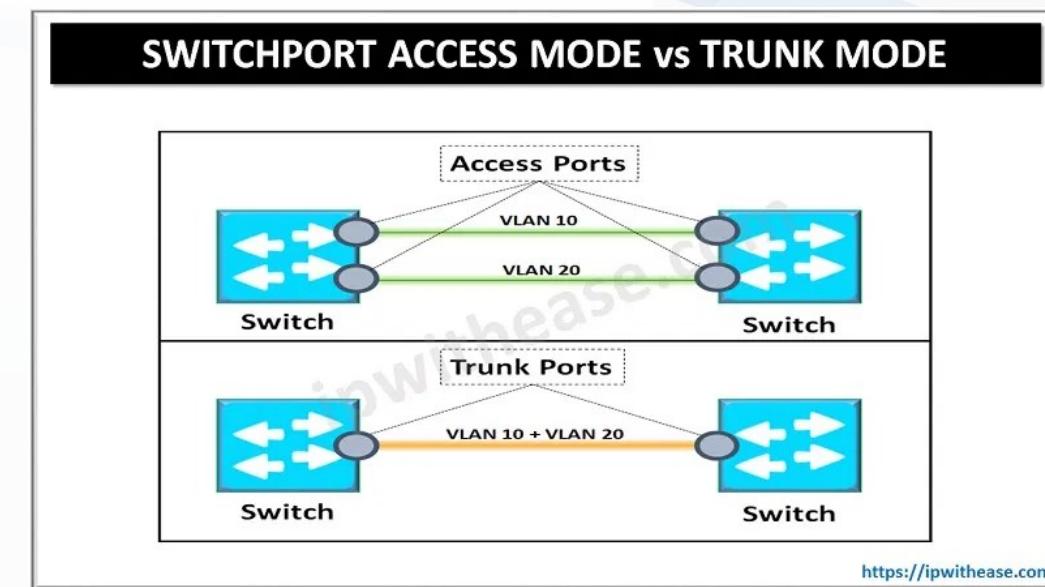
- Princip fungování
  - Zvolíme si kořen sítě Root Bridge – zařízení s nejnižší BID
    - BID = priorita + MAC
    - Root Bridge má všechny porty Designated a zároveň ve stavu Forwarding
    - Odešlu BPDU s vlastním BID
      - Pokud jsem nejnižší ostatní to akceptují
      - Pokud nejsem, stanice s nižším BID hodnotu změní a pošle dále
  - Dochází k rozpojení smyček – na základě rozeslaných a analyzovaných BPDU
  - Portům se nastaví jeden ze tří typů
    - Root port – port přímo spojený s Root switchem nebo s nejnižší cestou k němu, forward
    - Designated port – je členem STP, připojuje segment, forwardu
    - Non-designated port – blokovaný / alternativní port
  - Porty prochází více stavů
    - Blocking
      - Blokován, neposílá nic
    - Listenig
      - Naslouchá a přijímá BPDU rámce
    - Learning
      - Učí se MAC adresy, předává BPDU rámce
    - Forwarding
      - Plnohodnotný provoz



# Virtual LAN - VLAN, Trunk ( opakování z první přednášky )

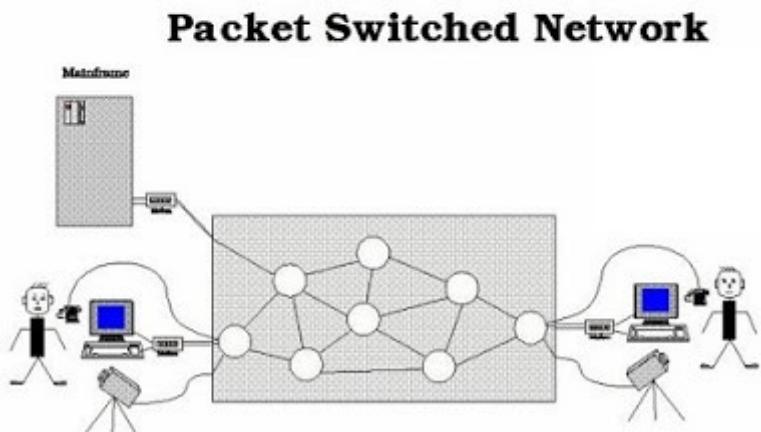
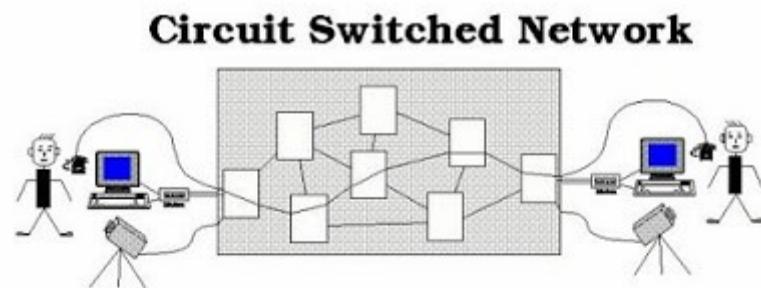


- VLAN – virtuální síť – virtuální rozdělení fyzického zařízení
- Dva typy portů:
  - VLAN virtuální LAN
    - „rozdělení“ jednoho fyzického zařízení na více virtuálních
    - Provoz v každé VLAN je isolované == nevidí data jiné VLAN
    - Identifikace VLAN pomocí VLAN ID, celé kladné číslo, výchozí je 1
    - V rámci jedné VLAN není nutné VLAN ID uvádět == „NETAGOVANÝ“ provoz
    - Porty v jedné VLAN se označují jako „access“ - „accessové“ porty
  - Trunk
    - Tím že se VLANy nevidí mezi sebou, musela by pro každou VLAN být samostatná cesta k dalšímu zařízení – switch / router
    - To je problém, protože pro 20 VLAN by se obsadilo 20 portů switche
    - Trunk je speciální port, kterým může procházet provoz více VLAN
    - Jednotlivé rámce se rozeznají pomocí VLAN ID v hlavičce rámce == „TAGOVANÝ“ provoz
    - Porty kde je povolen trunk jsou označované jako „trunkové“ porty



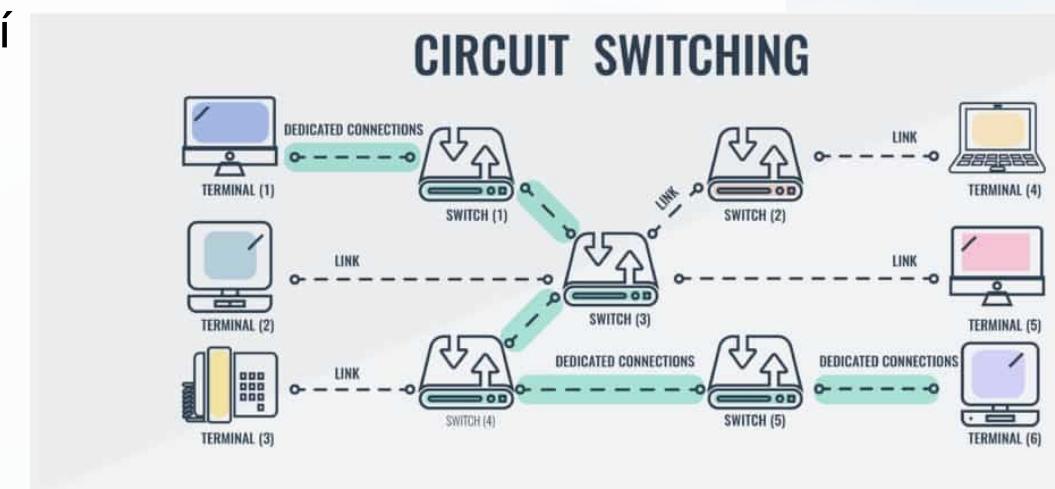
# Sítě dle typu přepínání

- Sítě může dělit dle toho jak jsou v nich data směrována
  - Zda je směrování řešeno jako :
    - Posílání bloků
      - Řešíme každý blok dat „samostatně“
    - Posílání streamu data
      - Připravíme cestu a tou pak posíláme všechna data
  - Základní možnosti
    - Sítě s přepínáním okruhů
      - Může se řešit i na L2
    - Sítě s přepínáním paketů / rámců
      - Může se řešit i na L2
    - Sítě s přepínáním zpráv



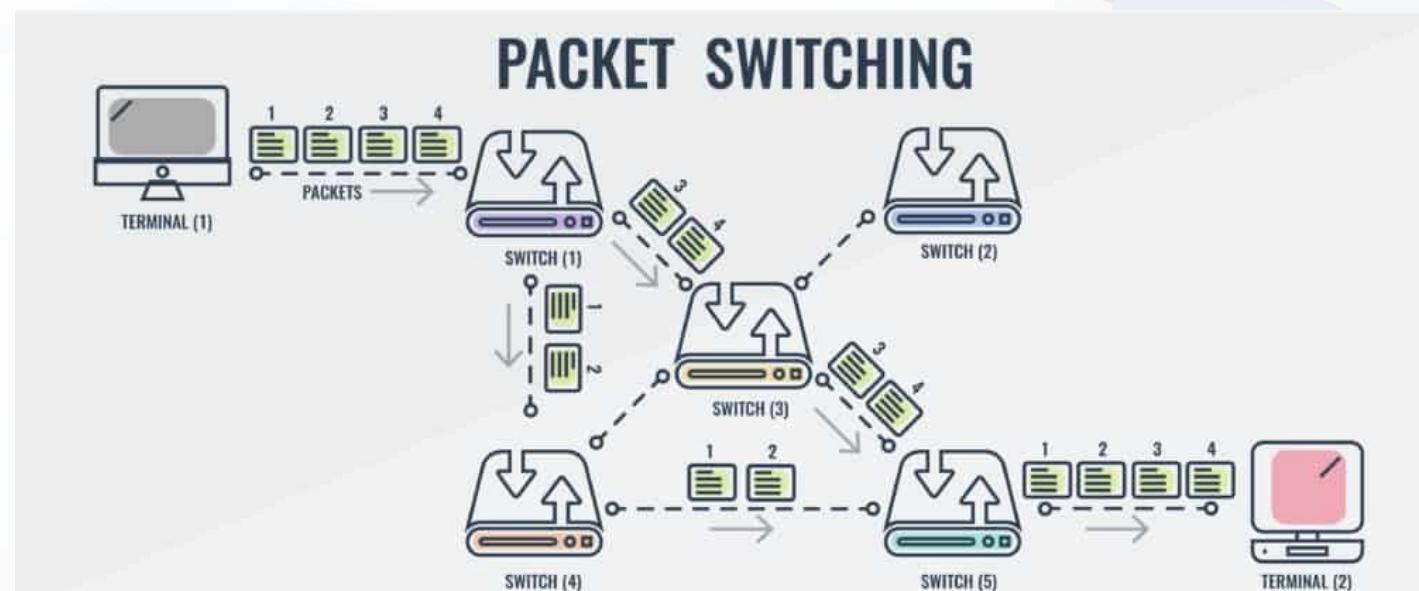
# Sítě dle typu přepínání: S přepínáním okruhů

- Hledání cesty je realizováno na začátku vysílání
- Je sestavena virtuální cesta pro daný přenos a tou je přenos následně realizován
- Řešení je pomalejší na začátku – musí se počkat než se cesta najde
  - Hledání se realizuje na základě žádosti o vysílání
- Vzniká problém v případě výpadku či změny v síti – je třeba virtuální cestu znova sestavit
  - Čím více výpadků a přepočítávání tím hůře
- Velkou výhodou je pak ale rychlosť následného zpracování
  - Data už se posílají po předem známé cestě a není nutné se v každém uzlu rozhodovat o cestě
  - Datová cesta je během přenosu vyhrazena pro daný přenos
- Jedná se o spojovaně orientovaný přenos
- Typicky se využívá v telefonní síti u hlasových služeb
  - Dříve i manuálně realizovaný přes spojovatelku



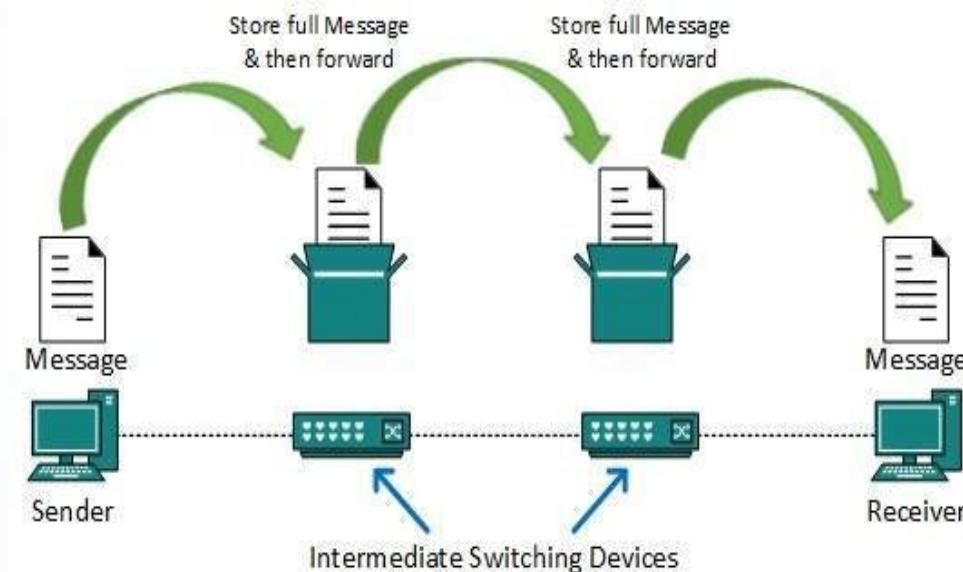
# Sítě dle typu přepínání: S přepínáním paketů

- Při realizaci přepínání okruhů je problém s časem nutným na přepočet při změně topologie či stavu sítě
  - Může dojít k odpojení linek nebo k jejich zahlcení
- Pokud ke změnám dochází často, bude sestavování cesty neefektivní
- Při přepínání paketů se rozhoduje na každém routeru
  - Rozhodování je samostatné
  - Nezáleží na předchozích odeslaných datech
  - Data mohou jít vždy různou cestou
  - V rámci dat musí být celá adresa příjemce
- Při intenzivním přenosu dat na stabilní síti bude mít vyšší řeži než sítě s přepínáním okruhů
- Dokáže se vypořádat s častými změnami v síti
- Dokáže optimalizovat přenos
  - Rozklad po více cestách



# Sítě dle typu přepínání: S přepínáním zpráv

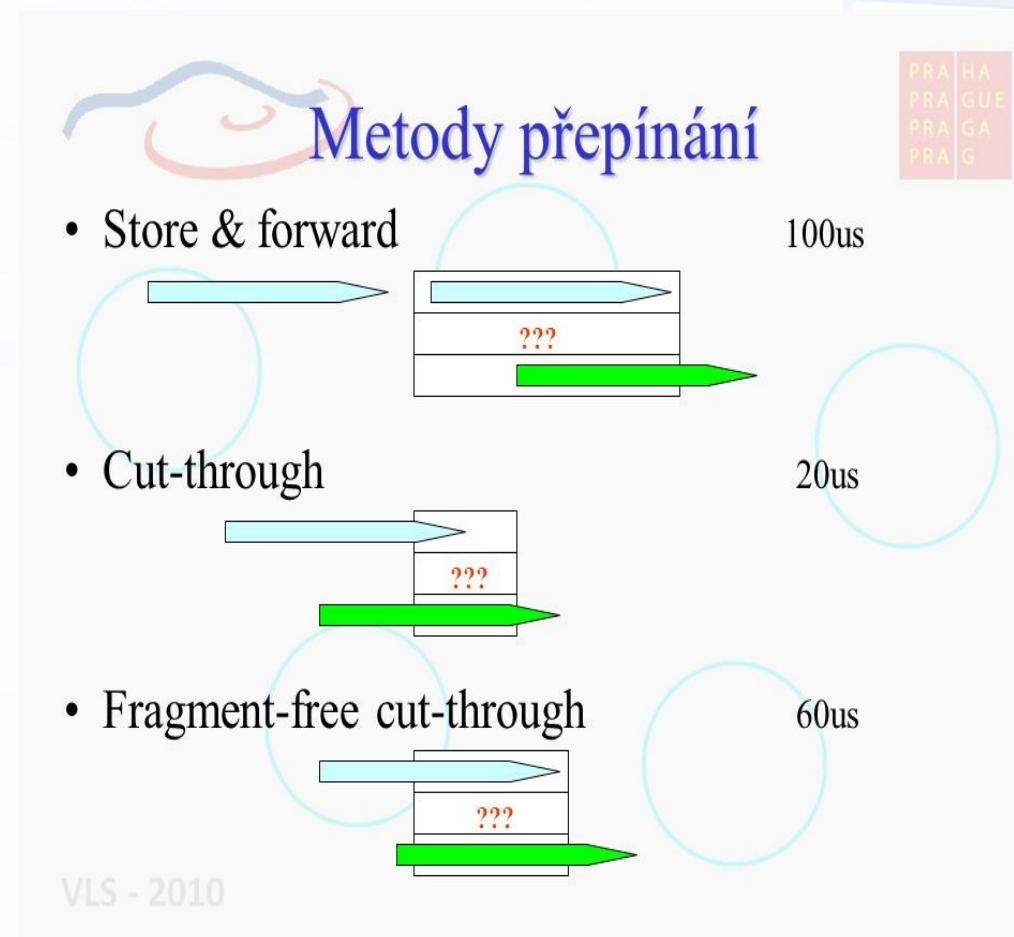
- Jedná se o „speciální“ případ sítí s přepínáním paketů
- Přenášenou jednotkou není paket, ale zpráva
  - Například email pro L7
- Princip přenosu je ale stejný jako u paketů, jen s jinou jednotkou
- Zpráva je přenesena na další uzel - celá – tam zkонтrolována a pokud je v pořádku je přeposlána dále
- Pokud v pořádku není, je tato informace zachycena na prvním směrovači kde došlo k chybě
  - Šetříme tím čas přenosu, protože v co nejkratší době reagujeme na problém
  - Typicky řešeno na vyšších vrstvách
    - Viz email, který prochází jednotlivými SMTP servery a na každém je jako celek - email – zpracován, doplněn o další data a poslán dále



zdroj: [https://www.tutorialspoint.com/data\\_communication\\_computer\\_network/physical\\_layer\\_switching.htm](https://www.tutorialspoint.com/data_communication_computer_network/physical_layer_switching.htm)

# Metod posílání dat

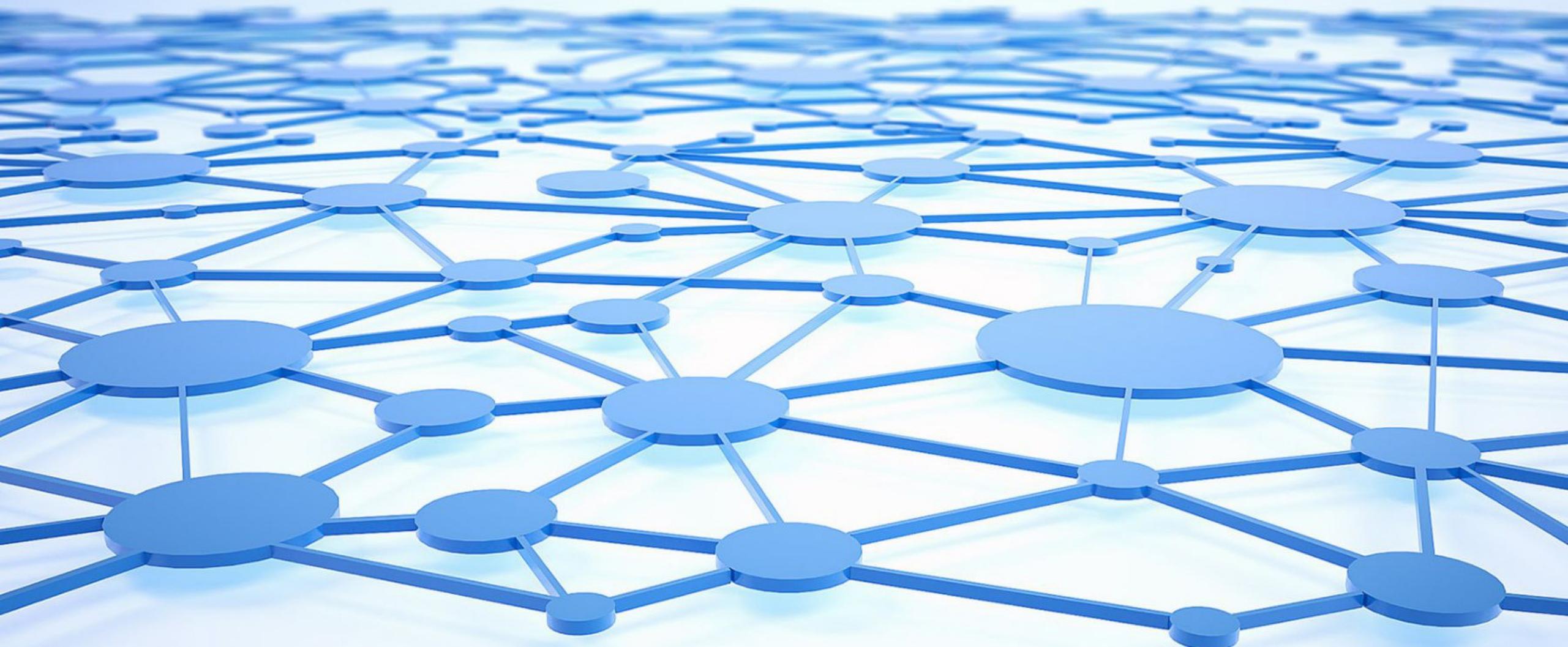
- Store & Forward
  - Přijme celý rámec / paket a teprve po jeho přijetí jej jako celek zpracovává
  - Může vyvažovat výkyvy v síti pomocí cache
  - Dochází k prodlevám, protože musí přjmout celý rámec / paket
- Fast forward / Cut-through
  - Začne odesílat data ihned po obdržení celé adresy příjemce
    - Adresa příjemce je na začátku data
    - Snižuje latenci
- Fragment-free
  - Kombinace obou předchozích
  - Nejprve přijme příjme data po adresu odesílatele, ověří že nenastal problém a pak posílá data dále



# Úvod do počítačových sítí

Přednáška 8 ( 2025/2026 )

ver. 2025-10-28-01



# L3 - Sítová vrstva

- Základní funkce
  - Hledání cesty – směrování – routing
    - Zjistit kam se mají data poslat
  - Předávání dat – forwarding
    - Když už vím kam data patří, tak je předám zvoleným směrem
- Rozšiřující / nepovinné funkce
  - Předcházení zahlcení
    - Snaha upravit provoz / směrování tak, aby se zahlcení předešlo
    - Protože pokud už k němu dojde, nejde dělat nic jiného než data zahazovat, což vede k nutnosti znova poslání a tím znova zatížení sítě
  - Řízení toku
    - Snaha předejít tomu, aby se zahltil příjemce – tedy síť je ok, ale nestihá příjemce
    - Výsledné chování může být velice podobné jako zahlcení sítě, ale má jinou příčinu a i řešení
      - Pokud je zahlcena síť, můžeme za určitých okolností posílat data jinudy/rozložit tok mezi více zařízení, ale pokud je zahlcen klient můžeme jen zpomalit vysílání
  - QOS – Quality of Service
    - Zajištění minimálně požadovaných zdrojů pro vybrané služby
      - Například pro hlas / multimédia – nepotřebuji moc, ale potřebuji pravidelně

# Spojení a stabilita

- V rámci L3 předpokládáme sítě s přepínáním paketů, které používají přenos dat metodou Store&Forward a můžeme realizovat jako službu/spojení:
  - Spojovanou
    - Pak přenášený blok dat označuje jako Paket
      - Obecněji je ale jako paket nazýván jakýkoliv blok dat naformátovaný jako paket
    - Před přenosem je jednorázově nalezena cesta a jako adresa je pak přenášen jen identifikátor cesty
    - Obdobně jako přepojování okruhů, ale cesta není vyhrazena, jedná se o virtuální okruh
    - Jednotlivé prvky – routery – si pamatují výsledek přešlého hledání
    - Například u ATM
  - Nespojovanou
    - Pak se přenášený blok označuje jako Datagram
    - Přenáší se celá adresa příjemce
    - K rozhodování o směrování dochází na každém uzlu / routeru
    - Například IP
- Přenos můžeme dělit i z pohledu spolehlivosti
  - Spolehlivý
    - Potvrzovaný – víme, že data došla
    - Typicky pro spojované protokoly
  - Nespolehlivý
    - Nepotvrzovaný – data odešleme, ale nemáme zajistěno, zda data dojdou a zda se o tom dozvíme
    - Pro spojované i nespojované protokoly
- Z výše uvedeného by to vypadlo, že nad IP( nespojovaná služba ) nejde realizovat spolehlivý přenos – to není zcela pravda, protože TCP spolehlivý je a je zároveň i spojovaný – ALE řeší se až na L4

# Směrovací tabulka

- Informace i možnostech směrování se ukládají do směrovací tabulky
- V tabulce jsou uloženy VŠECHNY dostupné cesty
- Logicky v tabulce nemohou být cesty ke všem strojům v síti
  - Jeden problém je že tabulka by byla obrovská
  - Druhý problém, že všechny cesty neznáme
- Náhradou cest o kterých nevím je výchozí cesta
  - Default gateway – pokud nevím kam, použiji toto pravidlo
- Cíle ve směrovací tabulce mohou být násobné
  - Tedy k jednomu cíli vede více cest
- Typy záznamů ve směrovací tabulce
  - Directly connect – cesty vzniklé z lokálně připojených sítí
    - Nejvyšší váha – jsem součástí dané sítě, takže mohu této informaci věřit
  - Static – ručně zadané cesty
    - druhá nejvyšší váha – administrátor „ví co dělá“
  - Dynamic – cesty vložené dynamickými směrovacími protokoly
    - Tyto cesty mohou být vložená přímo nebo zprostředkováně pomocí redistribuce

```
R1# show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
C   172.16.1.0/24 is directly connected, GigabitEthernet0/0
L   172.16.1.1/32 is directly connected, GigabitEthernet0/0
R   172.16.2.0/24 [120/1] via 209.165.200.226,00:00:12, Serial0/0/0
R   172.16.3.0/24 [120/2] via 209.165.200.226, 00:00:12, Serial0/0/0
R   172.16.4.0/28 [120/2] via 209.165.200.226, 00:00:12, Serial0/0/0
R   192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03, Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2 masks
C   209.165.200.224/30 is directly connected, Serial0/0/0
L   209.165.200.225/32 is directly connected, Serial0/0/0
R   209.165.200.228/30 [120/1] via 209.165.200.226, 00:00:12, Serial0/0/0
C   209.165.200.232/30 is directly connected, serial0/0/1
L   209.165.200.233/32 is directly connected, serial0/0/1
R1#
```

# Forwardovací tabulka: Administrative distance

- Pokud máme ve směrovací tabulce více cest se stejným cílem potřebujeme na základě „něčeho“ rozhodnout, kterou použít
- Toto rozhodnutí se dělá na základě „Administrative distance“
  - Jedná se o celé kladné číslo
  - Tato hodnota určuje váhu / důvěryhodnost dané informace
- Do forwardovací tabulky se ukládají informace s nejnižší dostupnou hodnotou AD pro danou cestu
- Důvěryhodnost protokolů je ve výchozím stavu dané, ale lze v případě potřeby i měnit
- Jednotlivé protokoly mohou mít různou důvěryhodnost dle místa použití
  - Interní EIGRP – 20
  - Externí EIGRP – 170
  - Jinak řečeno se každý protokol nehodí na všechna použití

```
Router_A#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       p - periodic downloaded static route

Gateway of last resort is not set

      10.0.0.0/24 is subnetted, 1 subnets
D  10.0.0.0 [90/30720] via 192.168.0.2, 00:00:09, FastEthernet0/0
C  192.168.0.0/24 is directly connected, FastEthernet0/0
Router_A#
```

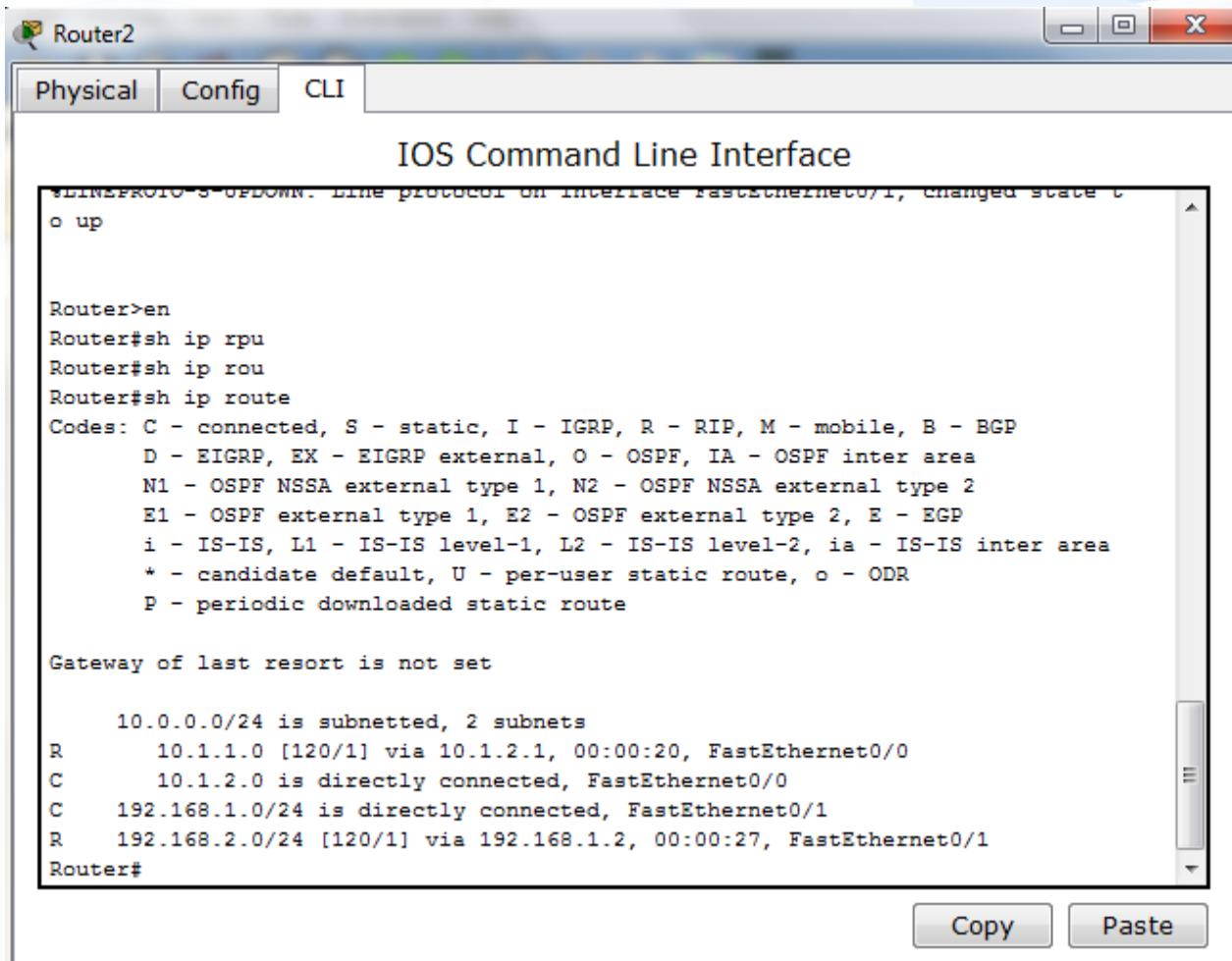
zdroj: <https://study-ccna.com/administrative-distance-metric/>

Routing Technique	Preference
Connected Interface	0
Static Route	1
EIGRP Summary Route	5
EBGP	20
Internal EIGRP	90
IGRP	100
OSPF	110
ISIS	115
RIP	120
EGP	140
ODR	160
External EIGRP	170
Internal BGP	200
Unknown	255

zdroj: <http://packetsanalyzed.blogspot.com/2013/04/hp-administrative-distance.html>

# Forwardovací tabulka

- Ve směrovací tabulce toho může být hodně
- Informace v ní mohou násobně
  - Tedy více cest jednomu cíli, ale my už potřebujeme data někam předat a je nutné říci kam
- K samotnému předání se pak použije forwardovací tabulka
- Jedná se podmnožinu informací ze směrovací tabulky
  - Pro každý cíl je jen jedna cesta
  - Je snaha počet řádek co nejvíce snížit
    - Například pomocí agregace
- Čím méně řádků v tabulce, tím rychlejší odbavení



The screenshot shows the Cisco IOS Command Line Interface (CLI) running on a router named 'Router2'. The window title is 'Router2' and the tab selected is 'CLI'. The interface displays the following output:

```
LINEPROTO-3-UPDOWN: Line protocol on interface FastEthernet0/1, changed state to up

Router>en
Router#sh ip rpu
Router#sh ip rou
Router#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
      i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
      * - candidate default, U - per-user static route, o - ODR
      p - periodic downloaded static route

Gateway of last resort is not set

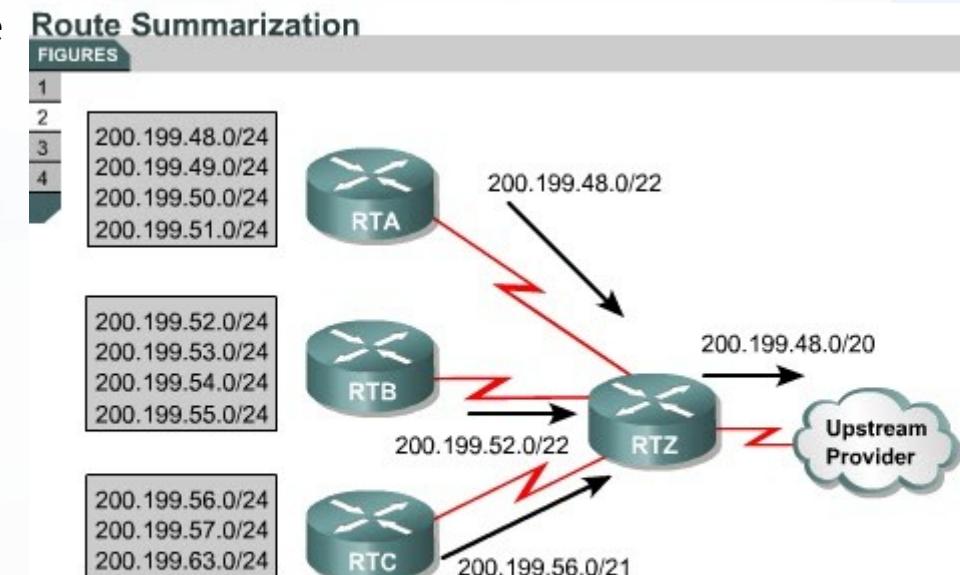
          10.0.0.0/24 is subnetted, 2 subnets
R        10.1.1.0 [120/1] via 10.1.2.1, 00:00:20, FastEthernet0/0
C        10.1.2.0 is directly connected, FastEthernet0/0
C        192.168.1.0/24 is directly connected, FastEthernet0/1
R        192.168.2.0/24 [120/1] via 192.168.1.2, 00:00:27, FastEthernet0/1
Router#
```

At the bottom right of the CLI window, there are 'Copy' and 'Paste' buttons.

zdroj: <https://community.cisco.com/t5/routing/unable-to-understand-this-routing-table-created-by-rip/td-p/1842945>

# Forwardovací tabulka: Agregace

- Kromě default gateway může počet řádek při směrování snížit agregace
- Cíl je z více řádků udělat jeden, který pokryje všechny původní řádky
  - Logickým požadavkem je, že odchozí port/IP musí být stejné
- Agregace může probíhat dvojím způsobem
  - Ručně – spojím vybrané sítě do jedné větší - viz obrázek
  - Automaticky – směrovač sám aggreguje jednotlivé cesty
    - Zde může nastat problém, že agregace může probíhat na úrovni tříd adres a nemusí být tedy žádoucí
    - Např. 10.0.0.0/24 a 10.1.0.0/24 se agregují na 10.0.0.0/8
      - Jedná se o třídu adres A, kde defaultní maska je /8
    - Automatická agregace lze na směrovačích vypnout



Route summarization reduces the routing table size by aggregating routes to multiple networks into one supernet.

zdroj: <http://basicitnetworking.blogspot.com/2012/11/route-aggregation-with-vlsm.html>

# Základní principy v směrování

- Směrování na základě cílové adresy / Destination base routing
  - Nejběžnější varianta směrování
  - Rozhodujícím kritériem je „cílová adresa“
    - Obsah ani zdroj data není relevantní pro rozhodování
  - Rozhodnutí nemusí ( ale může ) být vázáno na konkrétní adresu, ale spíše na síť do které adresa patří
    - Tedy se jedná o zobecnění ve snaze snížit počet záznamu ve směrovací / forwardovací tabulce
  - Dnes patrně nejrozšířenější metoda
- Směrování dle cesty s nejnižší „cenou“ / Least cost routing
  - Síť beme jako orientovaný ohodnocený graf, ve kterém hledáme nejlevnější cestu
    - Orientovaný – každá cesta nemusí být obou směrná
    - Ohodnocený – každá cesta má nějakou cenu ( latency, zatěžení, konstanty – např realně cena přenosu )
    - Klasická grafová úloha řešitelná například pomocí Dijkstrova algoritmu
- Směrování se řeší samostaně v každém uzlu – routeru / Hop by hop
  - Každý jeden uzel rozhoduje o dalším kroku – kam data předá
  - Rozhodování je samostatné, ale může vycházet ze společně získaných/sdílených informací
- Směrování je nezávislé na obsahu
  - Obsah dat / služba vyšších vrstev která jsou přenášena nemají na směrování vliv
    - Neplatí v případě QOS
- Směrování je bezstavové
  - Nezáleží na obsahu / cíly předchozího paketu při rozhodování o aktuálním

# Další možné principy v směrování

- Tyto principy nejsou obecně používané, ale mohou se v některých specifických situacích hodit
- Směrování podle zdrojové adresy / Source base routing
  - Je opakem k destination base routing
  - Může se destination base routingem kombinovat – napřed zkusím source base routing a když se pravidlo nenajde, použiji destination base routing
    - Je třeba doplnit rozhodovací pravidla – v Linuxu ip rules – který pomohou rozhodnout dle čeho se směruje
- Směrování na základě obsahu / Content switching
  - Může se někdy hodit mocí směrovat podle informací od vyšších vrstev - například L4
- Směrování se zohledněním historie / Flows
  - Pokud nějaká data patří k sobě – například stream multimédií, může být výhodné je posílat stejnou cestou

# Kategorizace směrování

- Dle reakce na změny
  - Neadaptivní
  - Adaptivní
- Dle metody řízení
  - Centralizované
  - Isolované
  - Distribuované
  - Hierarchické

# Kategorizace směrování: Dle reakce na změnu: Neadaptivní

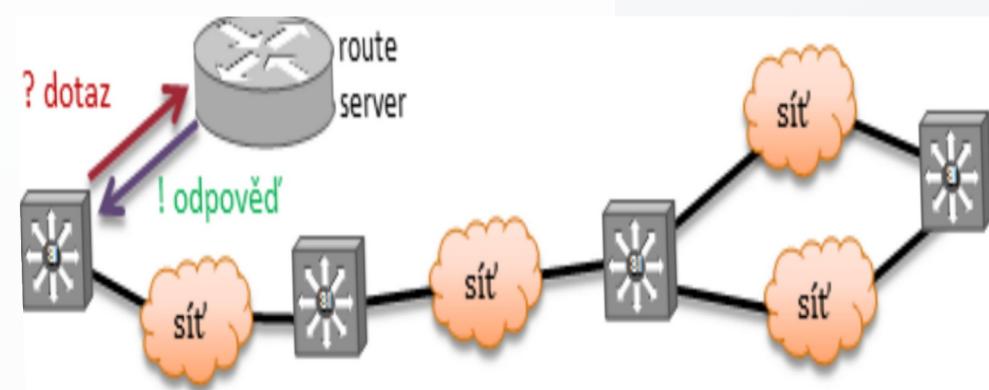
- Většinou se jedná o statické nastavení
- Nereaguje na změny v prostředí
  - Pokud je změna třeba – například při výpadku směrovače – musí někdo přijít a udělat změnu ručně
- Výhodou je vysoká míra predikovatelnosti
  - Bez ohledu nato co se děje, víte kudy vám data tečou
    - Což s ohledem na fakt, že jednotlivé spoje mohou mít různou cenu za přenos může být výhodné
  - Bezpečné řešení
    - Změny neprobíhají na základě stavu sítě, tedy nejde se směrováním manipulovat
    - Nemá žádnou přidanou režii z hlediska přenosu

# Kategorizace směrování: Dle reakce na změnu: Adaptivní

- Řeší nedostatky neadaptivního směrování
  - Tento požadavek vznikl „časem“ - u prvních sítě s nízkým využitím a malým počtem uzel nebyl nutný
- Adaptivní směrování se cyklicky snaží zjišťovat stav sítě a tyto informace pak promítnout do směrování
  - Reaguje na výpadek či přidání cesty
  - Může reagovat i na zhoršení parametrů existující cesty
    - Ke zhoršení může dojít vlivem vnějších vlivů – například interference nebo počasí, ale i vliv zatížení spoje – saturace linky
- Nevýhodou je nenulová režie adaptivních protokolů
  - Informace nutné k nastavení směrování se přenáší jako další data
    - Snižuje využití kapacity přenosového kanálu
  - Režie je tím větší čím více uzel a změn je v síti
  - Vzniká bezpečnostní riziko spojené s možností manipulovat s dynamickými směrovacími protokoly
    - V podstatě se jedná o snahu odklonit provoz jinam než kam správně patří z důvodů:
      - Možnosti zachytávání a případně modifikace dat – Man in the Middle
      - Znepřístupnění služby / sítě – Denial of Service

# Kategorizace směrování: Dle metod rozhodování: Centralizované

- Nejjednodušší cesta z pohledu implementace
- Máme jednu stanici, která řídí obsah směrovacích tabulek jednotlivých uzlů / směrovačů
  - Označuje se jako **route server**
- Ostatní stanice pouze realizují forwarding provozu na základě směrovacích informací od route serveru
  - Označované např jako **edge device**
- Výhodou je komplexní pohled na síť
  - Všechny informace jsou na jednom místě
  - Všechny informace se předávají v síti jen na jednu stanici
  - Je velice snadné, jako u všech centralizovaných řešení, měnit algoritmy
- Tak jako u všech centralizovaných řešení je problém výpadek route serveru
  - Single point of failure

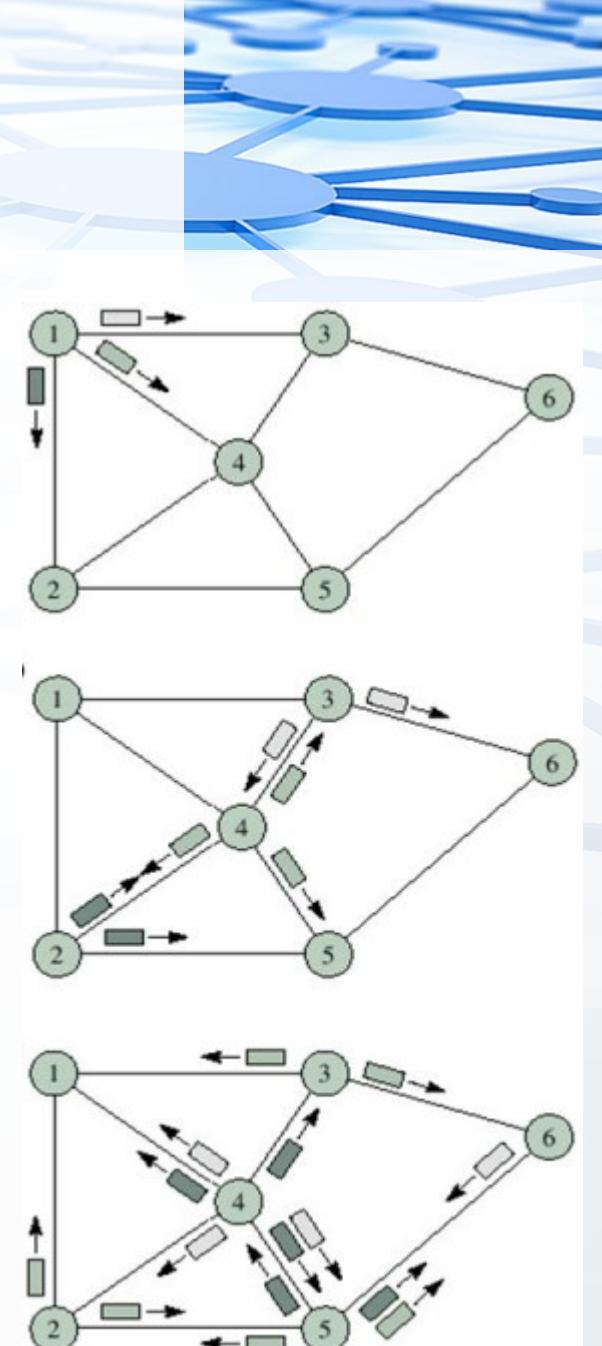


# Kategorizace směrování: Dle metod rozhodování: Izolované

- Každý uzel funguje autonomně a s ostatními nespolupracuje a ani není nikým řízen
- Data ostatním posílá a přijímá je, ale jen jako data, nikoliv jako řídící informace
- Existuje více metod izolovaného směrování
  - Záplavové směrování
  - „Horká brambora“
  - Náhodné směrování
  - Zpětné učení
  - Source routing
  - Policy base routing
- Nejsou tak masivně využívané samostatně, ale mají svůj význam ve speciálních případech
  - slouží jako řešení krizových či počátečních situací
  - Krizová – zařízení přestává stačit rozhodovat o směrování
    - Mohu použít náhodné směrování či metodu horké brambory
  - Počáteční – potřebuji nějakou informaci zjistit, ale zatím nemám jak
    - Použiji například záplavové směrování, protože pokud cesta existuje – najde ji

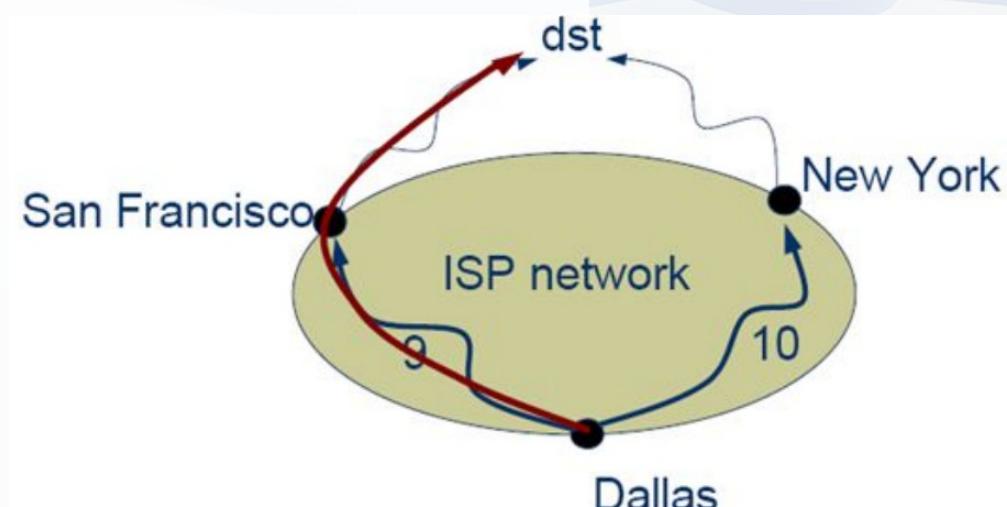
# Kategorizace směrování: Dle metod rozhodování: Isolované: Záplavové směrování

- Někdy označované jako Flooding
- Podobně jako u broadcastu pro switch jsou data rozeslána na všechny porty, krom toho ze kterého dorazila
  - Nepotřebuje žádnou směrovací tabulku
- Logickým důsledkem je fakt, že pokud nějaká cesta existuje, je nalezena a data doručena
- Nevýhod je ale více:
  - Pokud jsou v síti smyčky vniká problém násobných paketů
    - Stejně jako pro switch – broadcastová bouře
    - Násobné pakety je třeba nalézt a odstranit z provozu
    - Celá síť je násobně zatížena – nehodí se na běžný intenzivní provoz
  - Může být použit jako nástroj pro hledání a sestavení virtuální cesty
  - Může být použit tam, kde bez ohledu na režii potřebujeme mít jistotu, že data nakonec dojdou



# Kategorizace směrování: Dle metod rozhodování: Isolované: „Horká brambora“

- Metoda „horké brambory“ - Hot Potato
- Cílem je přicházející data co nejrychleji odbavit bez ohledu na vše ostatní
- Data se neodesírají podle směrovací tabulky, ale dle toho kterým portem mohou nejrychleji odejít
  - Což lze zjistit na základě délky výstupního bufferu jednotlivých portů
  - Což samozřejmě nemusí nutně být ten ideální nebo správný port
- Může mít dvojí reálné využití
  - Jako doplněk jiného „klasického“ směrování, kdy se na toto přepne v případě potíží
    - Na routeru se v bufferech hromadí data a router přestává stíhat, přepne na tuto metodu a v co nejkratším čase fronty vyprázdní nebo alespoň výrazně sníží jejich délku.  
A to i za cenu toho, že některá data nemusí nutně dojít k cíli a už vůbec ne ideální cestu => přehodím ten problém „tu horkou bramboru“ na někoho dalšího
- Pro vyvažování zátěže násobných linek
  - Pokud mám dvě či více cesty z bodu A do bodu B, mohu jejich provoz pomocí této metody optimalizovat/vyvažovat
    - Protože jak se řeklo, aktuální paket odejde cestou s nejkratším bufferem

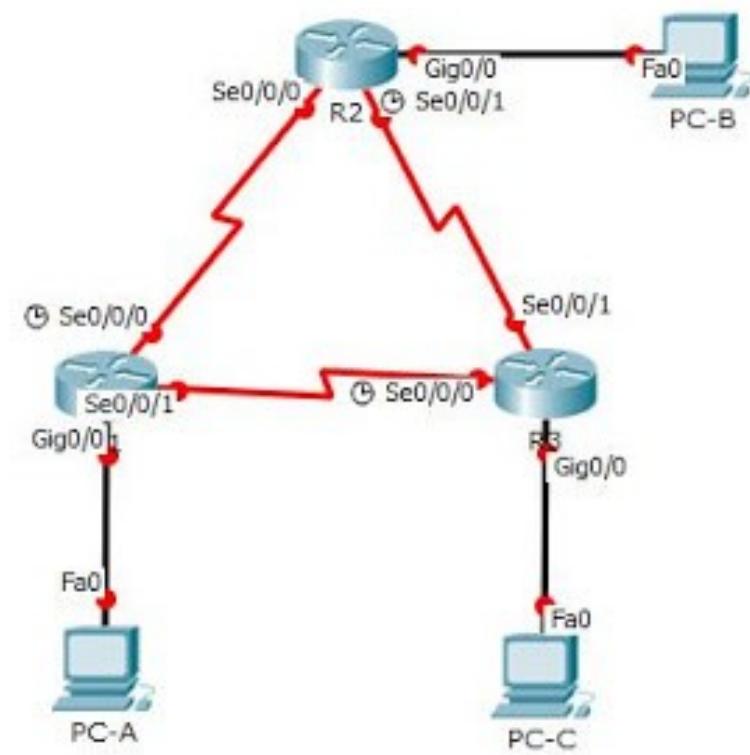


# Kategorizace směrování: Dle metod rozhodování: Isolované: Náhodné směrování

- Náhodné směrování - Random walk
- Jak už název napovídá jedná se o náhodné směrování
- Směr / port není volen na základě nějakého kritéria, ale náhodně
  - Například na rozdíl od „horké brambory“, kde sice také není směrovací tabulka, ale nějaký princip ano
- Používá se v situacích kdy na rozhodování není čas / zdroje / důvod
  - Například při zahlcení směrovače toto zařízení typicky data zahazuje, s použitím náhodného směrování se počítá s dvojím možným benefitem
    - Jenak i náhodně mohu trefit tu správnou cestu – šance není velká – podle počtu portů – ale je nějaká
    - I pokud netrefím správný směr, je pořád možné, že další směrovač na tom bude lepé a bude vědět kam data poslat
      - Takže sice o jeden či více kroků cestu prodloužím, ale zabráním ztrátě dat, timeoutu a znova poslání
- Reálně se používá například v senzorických nebo bezdrátových sítích

# Kategorizace směrování: Dle metod rozhodování: Isolované: Zpětné učení

- Zpětné učení - Backward learning
- Používá směrovací tabulku, kterou si postupně plní
  - Na počátku je prázdná – nepotřebuje počáteční informace
  - Pokud přijde paket od A pro B, poznačí si do směrovací tabulky cestu k A
  - Data mají jít k B, ale neví se kudy, takže není na výběr a data pošleme na všechny ostatní porty daného routeru
    - Použije se záplavové směrování jako „berlička“ k nalezení cesty, protože pokud existuje, záplavové směrování ji najde
  - Pokud data dorazí až do B a ten pošle odpověď – kam ví, protože to se naučil na základě příchozích dat – naučí se z odpovědi první router i cestu k B
- Výhodou je, že nepotřebuji žádnou výchozí konfiguraci
- Nevýhoda je dlouhá doba a režie na učení
  - Ta je o to horší, čím více se síť mění v čase
  - S počtem komunikujících stanic narůstá i délka tabulek, což také není žádoucí
- V reálu se nepoužívá na L3, ale používá se tento princip na L2 v Ethernetu
  - Pro mosty/přepínače kde data chodí jen ne konkrétní porty, na rozdíl od hubu

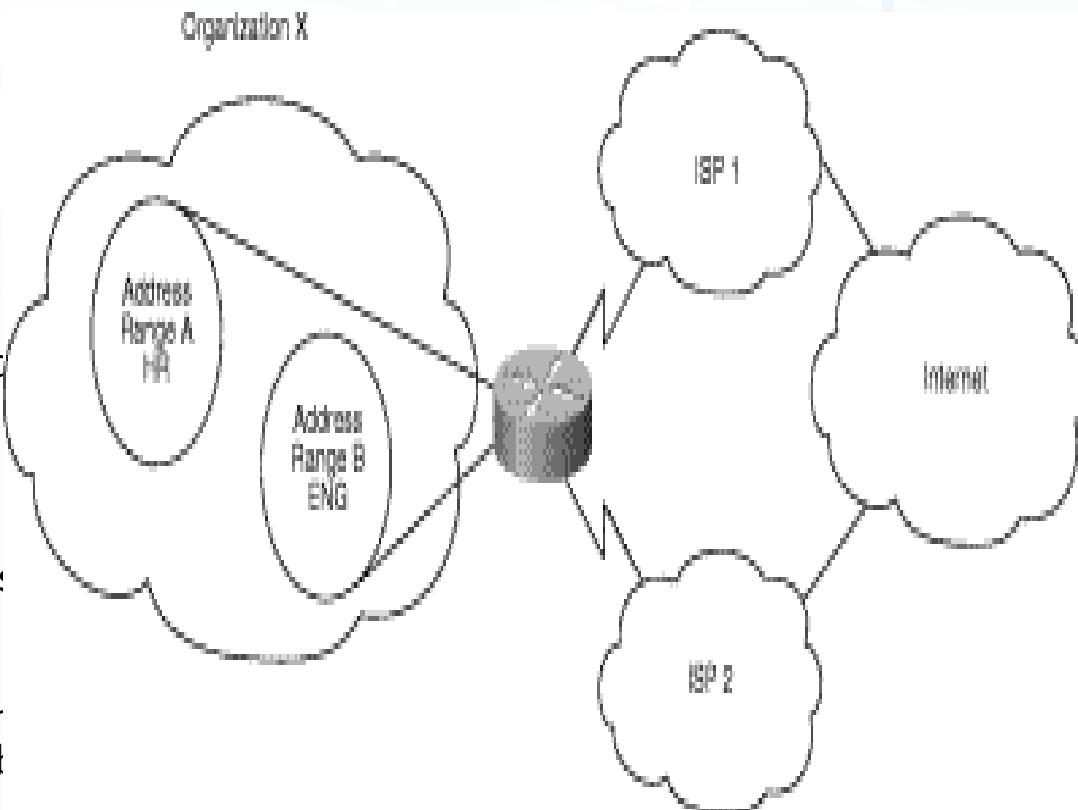


# Kategorizace směrování: Dle metod rozhodování: Isolované: Source routing

- Směrování od zdroje – Source routing
- Kudy data půjdou určuje odesílatel tím, že tyté informace vloží do hlavičky paketu
  - Tedy ne jen adresa cíle, ale rovnou celá cesta
- Kudy mohou data jít odesílatel najprve zjistí pomocí záplavového směrování
  - Drobné modifikace je v tom, že záplavově směrovaný paket do sebe uchovává informaci o uzlech kterými prošel
  - Až dojde k cíli použije tuto sekvenci k návratu a zároveň tím zjistil cestu
- V praxi se opět na L3 nepoužívá
  - Je nutná podpora na směrovačích
  - Má vysokou režii
- Používá se ale na L2 v rámci Token Ringu
- **POZOR – neplést se Source base routingem**

# Kategorizace směrování: Dle metod rozhodování: Isolované: Policy base routing

- Směrování podle pravidel – Policy base routing
- Varianta směrování dle „dalších pravidel“
  - Zdrojová adresa
  - Port / protokol
  - Metadata jako je typ paketu, jeho velikost atd.
- Běžně se nepoužívá kvůli vyšší režii, ale má své využití v krajních situacích
  - Například při použití více směrovacích tabulek v jednom stroji na základě adresy zdroje
- Typicky PBR má vyšší prioritu než běžné směrovací tabulky
- Příklad:
  - firma X má nařízeno, že provoz z rozsahu IP adres A, je směrováno přes ISP 1 a provoz z rozsahu IP adres B je směrováno přes ISP 2
  - Tohle běžným routingem řešit nejde, protože by se vždy použila výchozí brána => potřebujeme PBR kde podle pravidla testující zdrojovou adresu může použít jinou směrovací tabulkou s jinou výchozí branou



zdroj:

<http://www.cs.vsb.cz/grygarek/SPS/projekty0405/RouteOptimization/dokumentace/ar01s02.html>

# Kategorizace směrování: Dle metod rozhodování: Distribuované

- Distribuované směrování není založené na samostatném směrování každého jednoho uzlu ani na jedné centrální autoritě, ale využívá společného algoritmu a předávání informací
- Jednotlivé uzly se vzájemně informují o dostupných sítích, o jejich parametrech atd. a na základě toho si každý uzel sestaví svoji směrovací tabulku
- Tyto algoritmy počítají se změnami v síti a informace průběžně nebo na základě změny aktualizují
  - Jedná se o adaptivní řešení
- Distribuované směrovací algoritmy dělí na :
  - Interní - IGP ( Interior gateway protokol )
    - Distance vector protokoly
    - Link state
  - Externí – EGP ( Exterior gateway protokol )
    - „Path vector“
- **POZOR: Dynamické směrovací protokoly provoz Nesměrují/Nepřenášejí data, ale jen plní směrovací tabulky, které se ke směrování následně používají**
- Z pohledu počítačové sítě se jedná o běžné aplikační protokoly

# Kategorizace směrování: Dle metod rozhodování: Distribuované: Metrika

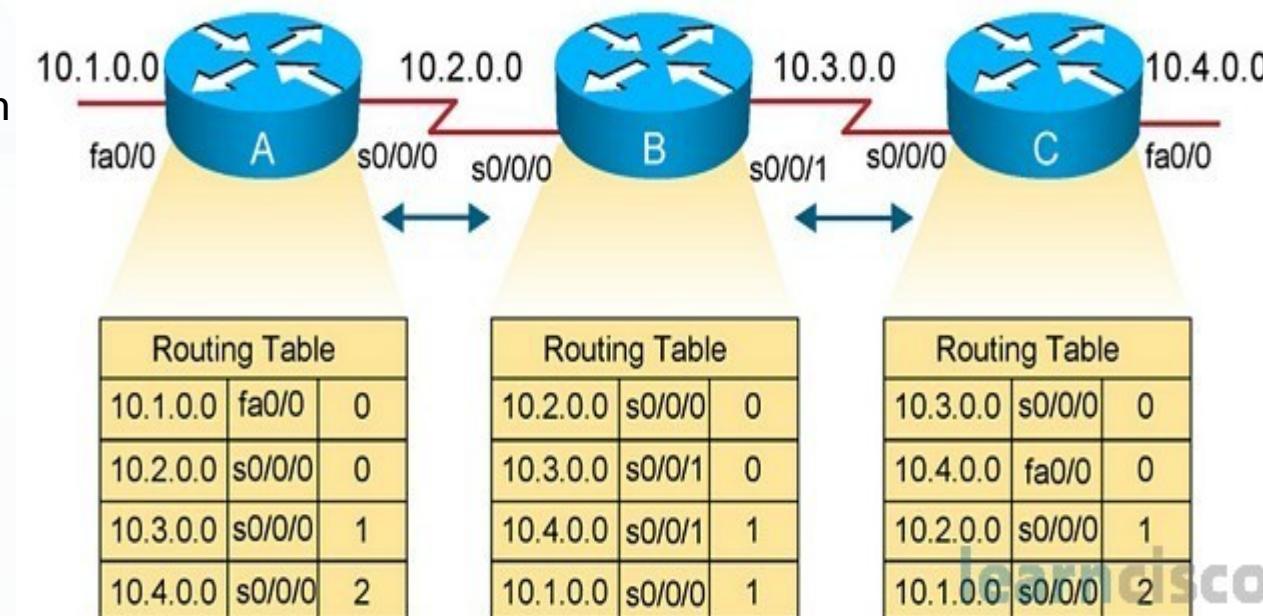
- Metrika je typicky číslo, které určuje „kvalitu“ dané cesty v rámci směrovacího protokolu
- Pokladem pro metriku může být jedna nebo i více veličin
- Tři nejběžnější modely:
  - Distance vector – metrikou je délka vektoru vzdáleností
    - Tedy např. přes kolik dalších routerů musí paket projít aby došel k cíli
  - Link state – metrikou je cost/cena, která se typicky určuje na základě rychlosti linek, tedy šířky pásma
  - Path vector – obdobně jako u distance vector se hledá nejkratší cesta z pohledu skoků, ale ne po jednotlivých routerech, ale po jednotlivých autonomních oblastech
- Metrika se používá k rozhodnutí, kterou z násobných linek ve směrovací tabulce promítnout do forwardovací tabulky
- Podle konkrétního protokolu může hodnota metriky označovat i nedostupnou cestu
  - Pro Distance vector protokol RIP je to například 16
    - 16-ctý uzel se bere jako nekonečno – jako by tam už ani cesta nebyla

# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP : Distance vector

- Distance vector protokoly – DVA nebo DVR
- Metrikou těchto protokolů je „vektor“ vzdáleností
  - Tedy např. počet mezilehlých směrovačů
- Neberou v potaz reálné parametry linek, ale jen „vzdálenost“ v podobě počtu uzlů
  - Reálně tedy mohou data poslat kratší, ale mnohem pomalejší cestou
- Výhodou je snadné zjišťování „stavu sítě“, neboť se realizuje jen jako výměna informací mezi dvěma sousedy
  - Tedy mají je připojené nebo je dostaly od jiných sousedů
  - Není třeba linky proměřovat, tedy nebereme v potaz reálné parametry linek mezi uzly ani jejich zatížení
- Každý uzel má jen částečnou informaci o stavu celé sítě
- Problém nastává s timeouty u velkých sítí
  - Musí být zvolena hranice, kdy se další uzel za už považe za nedostupný
  - Velikost sítí je tak reálně omezena dostupným počtem směrovačů v řadě
- DVA algoritmy pomalu konvergují
  - Informace o výpadku se šíří pomalu, protože v každém kroku se informace výpadku předá jen sousedům

# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Distance vector: Tvorba tabulky

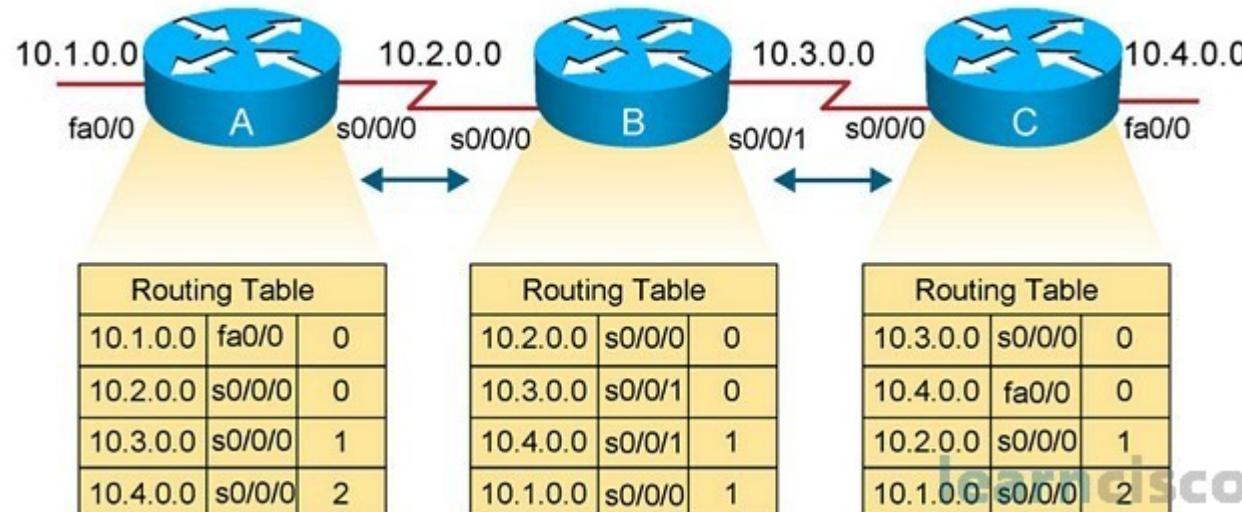
- Postupná tvorba směrovací tabulky
  - V prvním kroku znám jen své připojení sítě, které inzeruji/posílám přímým sousedům
    - Pro A 10.1.0.0 a 10.2.0.0 s metrikou nula
    - Pro B 10.2.0.0 a 10.3.0.0 s metrikou nula
    - Pro C 10.3.0.0 a 10.4.0.0 s metrikou nula
  - V druhém kroku dostanu info od svých sousedů a ty doplním do své tabulky
    - Pro A
      - 10.1.0.0 a 10.2.0.0 s metrikou 0
      - 10.3.0.0 s metrikou 1 ( od B )
    - Pro B
      - 10.2.0.0 a 10.3.0.0 s metrikou 0
      - 10.1.0.0 s metrikou 1 ( od A )
      - 10.4.0.0 s metrikou 1 ( od C )
    - Pro C
      - 10.3.0.0 a 10.4.0.0 s metrikou 0
      - 10.2.0.0 s metrikou 1 ( od B )



zdroj: <https://www.learnCisco.net/courses/icnd-1/ip-routing-technologies/enabling-rip.html>

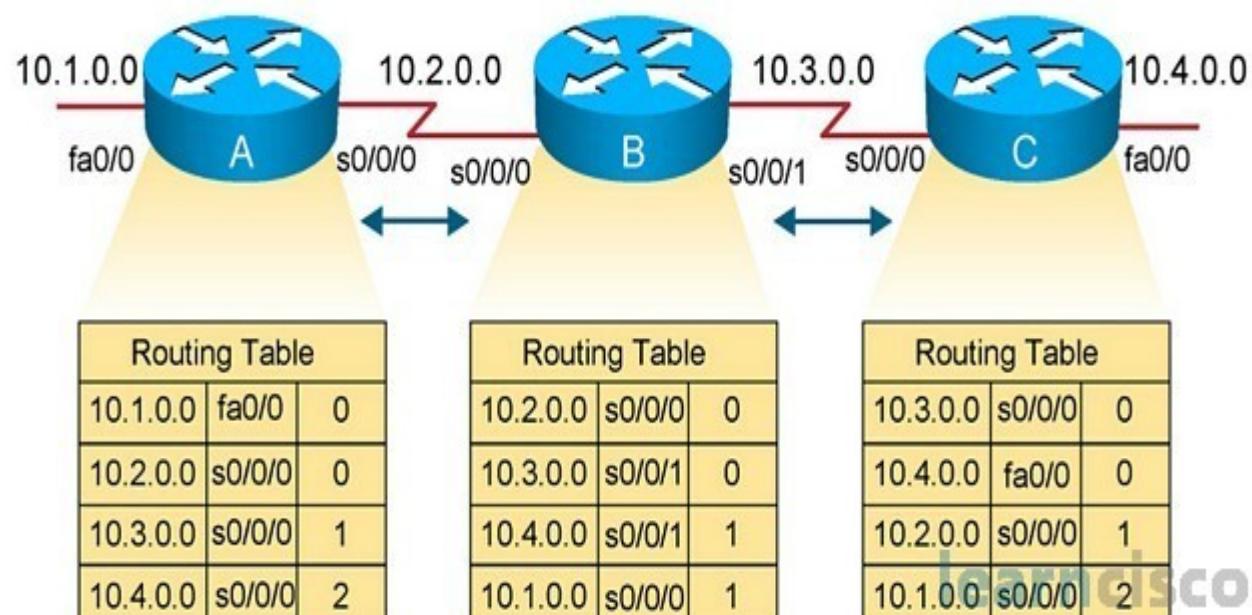
# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Distance vector: Tvorba tabulky II.

- Postupná tvorba směrovací tabulky
  - V třetím kroku
    - Pro A
      - 10.1.0.0 a 10.2.0.0 s metrikou 0
      - 10.3.0.0 s metrikou 1 ( od B )
      - 10.4.0.0 s metrikou 2 ( od B, původně od C )
    - Pro B
      - 10.2.0.0 a 10.3.0.0 s metrikou 0
      - 10.1.0.0 s metrikou 1 ( od A )
      - 10.4.0.0 s metrikou 1 ( od C )
    - Pro C
      - 10.3.0.0 a 10.4.0.0 s metrikou 0
      - 10.2.0.0 s metrikou 1 ( od B )
      - 10.1.0.0 s metrikou 2 ( od B, původně od A )



# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Distance vector: Problém počítání do nekonečna

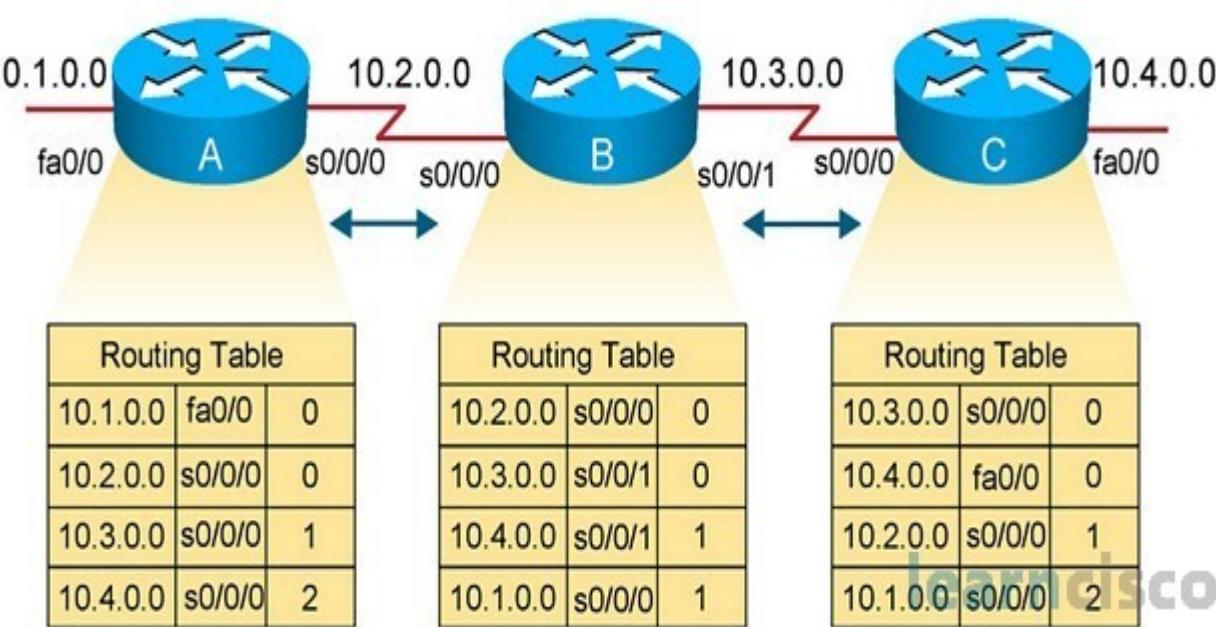
- Problém počítání do nekonečna – count of infinity
- Sousední routery si vyměňují informace včetně těch, které nejsou jejich lokální, ale naučily se je
- Problém nastává při výpadku, např. pro uzel A
  - Pokud je síť stabilní, má B dvě info k 10.1.0.0
    - Od A s cenou 1
    - Od C s cenou 2
      - Tu ale nepoužije, protože od A má lepší cenu
  - Pokud ale A vypadne, v B se použije cesta od C s hodnotou 2 a B si ji uloží s hodnotou 3 ( 2 od C + 1 cesta z B do C )
  - V dalším kroku C dostane info od B, že tuto síť umí s cenou 3 a upraví si svoji tabulkou na 4 ( 3 od B + 1 na cestu z C do B )
  - A info zas pošle dále ...
  - Problémů je zde více
    - Tohle by nikdy neskončilo – můžeme vyřešit limitem – např 16 pro RIP
    - Celé šíření výpadku velice dlouho trvá



zdroj: <https://www.learnncisco.net/courses/icnd-1/ip-routing-technologies/enabling-rip.html>

# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Distance vector: Řešení problému počítání do nekonečna

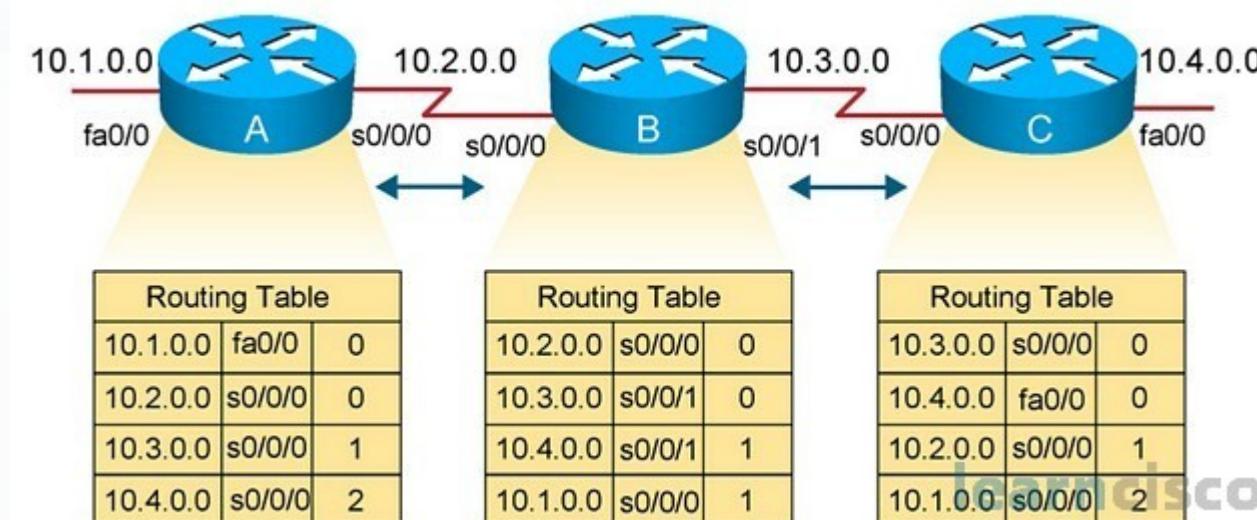
- Základní řešení se označuje jako „split horizon“ - rozštěpený horizont
  - Pokud se cestu do 10.1.0.0 dozvím od B( jako uzel C ) , tak už ji zpět do B neinzeruji
  - Tedy pokud vypadne uzel A, cesta zmizí z B a v dalším kroku i z C
- Rozšířením „split horizon“ je „split horizon with poisoned reverse“
  - Rozštěpený horizont s otráveným zpětným kanálem
  - Já sice tu cestu zpět inzerovat budu, ale s hodnotou nekonečno
- Důsledkem obou metod je rychlejší šíření negativní informace – info o výpadku cesty
- Problém částečně zůstává, neboť nejde zabránit vzniku cyklů
  - Nevracím data zpět, ale pokračuji pořád dál jedním směrem
  - Možným řešením je neaktualizovat data ihned, ale počkat na info od dalších uzlů, ale to bude zpomalovat konvergenci (Hold-Down)
    - Řešením by byl triggered update – okamžitá informace o změně
      - Pokud jsem upravil svou tabulku IHNED to info předám dále



zdroz: <https://www.learnCisco.net/courses/icnd1-ip-routing-technologies/enabling-rip.html>

# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Distance vector: RIP

- Nejtypičtějším DVA protokolem je RIP
- Metrikou pro RIP je jen počet kroků k dosažení požadované sítě
  - Počet mezilehlých směrovačů
- Informaci o dostupných sítích rozesílá každý uzel každých 30s
  - Jedná se o vlastní směrovací tabulku složenou s lokálně připojených sítí a získaných informací
- Pokud nové info nepřijde od souseda déle než 180s, je prohlášen za mrtvého
- RIP pro komunikaci používá 520/UDP
  - Jako běžný aplikacní protokol potřebuje prostup ve FW
- K vyhledávání používá Bellman-Fordův algoritmus
  - Hledání nejkratší cesty v ohodnoceném grafu
- Používá „Split horizon with poisoned reverse“ a „Triggered update“
- Existují tři verze:
  - RIP 1
    - Původní verze, neřeší zabezpečení, pracuje jen s třídami adres
    - Podporuje 15 uzlů v řadě, 16-tý se rovná nekonečnu – nedostupný
    - Data posílá broadcastem
  - RIP 2
    - Rozšiřuje původní verzi o podporu CIDR – pracuje s maskami sítí
    - Data posílá multicastem
    - Zavádí autentizaci heslem
  - RIP ng ( next generation )
    - Přináší podporu pro IPv6



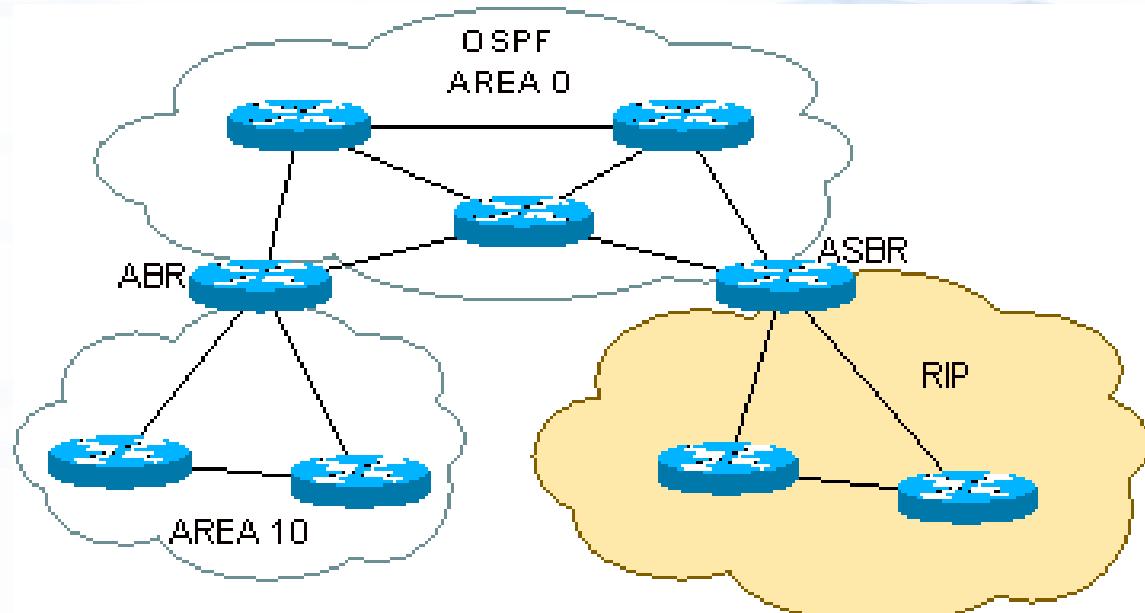
zdroj: <https://www.learnCisco.net/courses/icnd-1/ip-routing-technologies/enabling-rip.html>

# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Link state

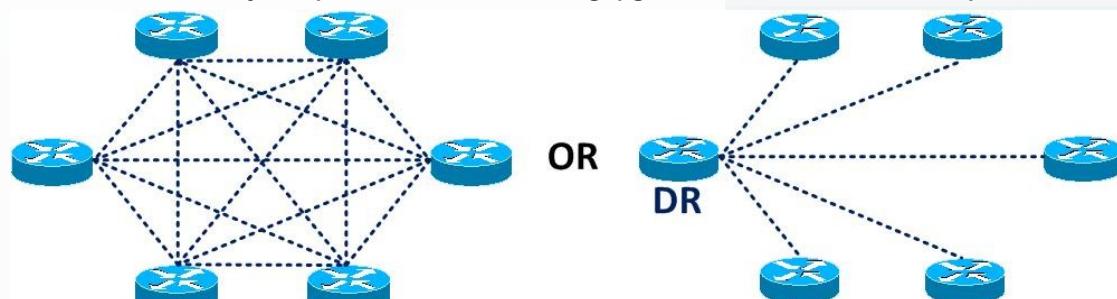
- Nedostatkem DVA je fakt, že nezohledňuje aktuální stav sítě ani její parametry
- Link state protokoly používají jako metriku cenu – cost, což je informace lince mezi uzly
  - Zda je linka 1Mbps, 10Mbps, ....
- Každý uzel cyklicky testuje dostupnost svých sousedů pomocí „Hello“ zpráv
  - Ví, že soused žije a ví jak „rychlá“ k němu vede linka
- Každý uzel šíří všem ostatním uzlům informace, které má
  - Zásadní rozdíl proti DVA – info už nepředávám jen sousedům, ale VŠEM
- Jednotlivé uzly si sami vypočítávají směrovací informace
  - Dostávám info od všech, takže vím o stavu celé sítě
  - Pokud jeden uzel udělá chybu ve výpočtu neovlivní to další uzly
- Informace o stavu linek – Link State pakety - LSP
  - Informace se musejí rozesílat při změně, ale VŠEM uzlům
    - Novou informaci mají všechny uzly v prvním kroku, což výrazně urychluje konvergenci oproti DVA
  - Informace se může rozesílat i periodicky, ale spíše pro osvěžení informací a za delší dobu než u DVA
    - Například jen jednou za 30min
  - Uzel, který přijme LSP, jej pošle všem sousedům, kromě toho od koho info dostal
  - V rámci LSP je identifikace verze – pokud příjmu LSP se starší verzí zahodím jej
  - Přenos LSP je spolehlivý
    - Využívá potvrzení doručení, timeouty a případné opakování zaslání informace
- Ve výsledku má LSA menší režii než DVA v závislosti na počtu uzel a proto se lépe hodí do rozsáhlejších sítí
- LSA rychleji konverguje a neobsahuje problém počítání do nekonečna

# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Link state: OSPF

- Dnes nejběžnější představil LSA protokolů
- Používá Dijkstruv algoritmus
- Cyklicky posílá sousedům Hello pakety pro zjištění jejich stavu
  - A díky tomu může detekovat změny, kde info o změně pak posílá všem
  - Posílá se každých 10s a posílá se přes multicast
  - Pokud do 40s neobdržím Hello, prohlásím souseda za mrtvého
- Podporuje autentizaci a summarizaci
- Pro velké sítě zavádí dělení na subsítě - area
  - Základní je Area0, ke které jsou připojeny další oblasti AreaX
  - Komunikaci mezi oblastmi zajišťují ABR směrovače
    - Mají přístup do obou sítí
    - Předávají JEN summarizované informace z dané sítě, ne celou topologii
  - Existují ještě ASBR směrovače, které jsou pouze transportní
    - Umožňují importovat routy z jiných systémů jako je např RIP
- Pro broadcastové sítě zavádí tři stavy routerů
  - DR - Designed router - pověřený router
    - Je volen na základě konfigurační konstanty a MAC adresy
  - BDR - Backup Designed router - záložní pověřený router
  - DRO - ostatní routery, které se připojují k DR nebo BDR
- Komunikace pak není každý s každým, čímž se šetří přenosy a zrychluje konvergence



zdroj: <http://www.cs.vsb.cz/grygarek/SPS/lect/OSPF/ospf.html>



zdroj: <https://jjrinehart.wordpress.com/2012/07/30/ospf-iv-there-is-no-i-in-team-more-about-drs/>

# Kategorizace směrování: Dle metod rozhodování: Distribuované: IGP: Link state: OSPF: Cena linek

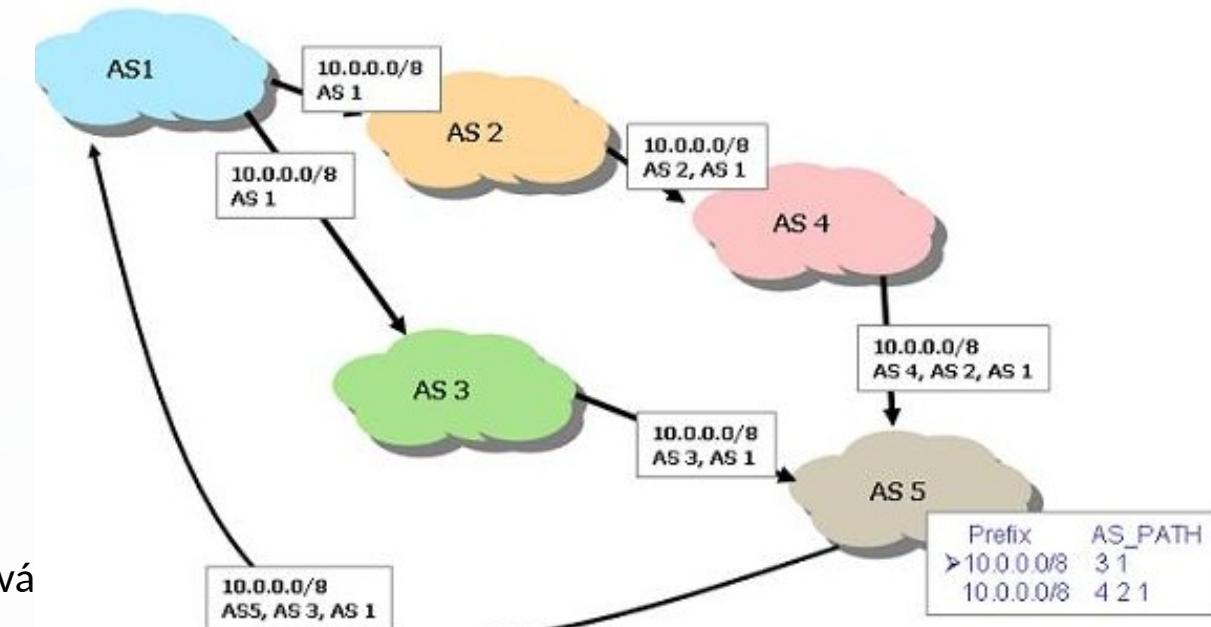
- Výchozí cena linky v OSPF je
  - Cena =  $100.000.000/\text{bandwith}$
  - Tedy reference k 100Mbps – vychází z historie, když se 100Mbps bral jako maximální možná hodnota
- Zde vzniká problém, protože od 100Mbps výše bude cena linek 1
  - Ale to neodpovídá, protože 100MBps není stejný jako 10Gbps
- V routerech jsou dvě možnosti řešení
  - Upravit referenční hodnotu výpočtu
    - Pro cisco např ospf auto-cost reference-bandwidth XXX
  - Nastavit cenu linky ručně
    - Pokud se například jedná jen o jednu linku v systému
    - Např pro cisco ip ospf cost XX

Auto-Cost Reference-Bandwidth 1000

Interface Type	Cost Value
10 Gigabit Ehternet (10 Gbps)	1
Gigabit Ethernet (1 Gbps)	1
Fast Ethernet (100 Mbps)	10
Ethernet (10 Mbps)	100
Serial (1.544 Mbps)	647
Serial (128 Kbps)	7812
Serial (64 Kbps)	15625

# Kategorizace směrování: Dle metod rozhodování: Distribuované: EGP: Path Vector

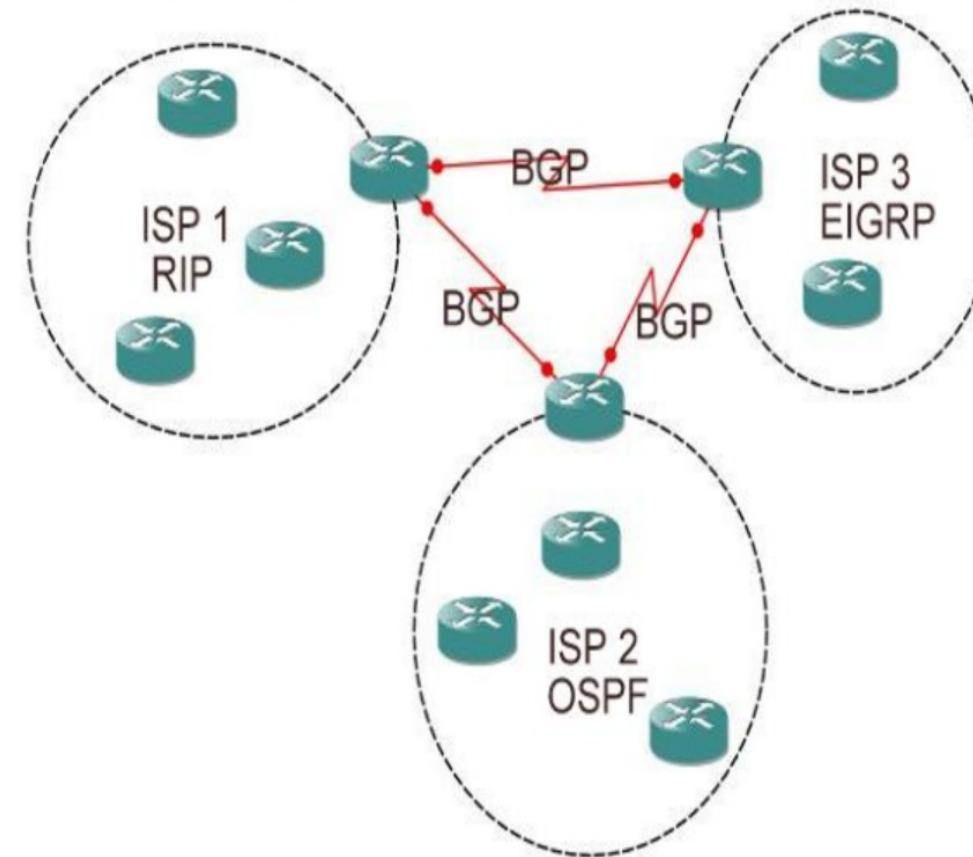
- Path vector se používá v externích dynamických směrovacích protokolech
- Kromě informace o cílové sítí obsahuje
  - Router pomocí kterého je dostupný
  - Celou cestu k cílové sítí
    - Cesta je definovaná jako seznam autonomních oblastí
- Někdy je zařazován mezi Distance vector protokoly
- Autonomní systém - AS
  - Samostatně spravovaná oblast s vlastním identifikátorem
  - Samostatná směrovací doména
    - Co se děje uvnitř se neprojevuje nahodile navenek, ale jen prostřednictvím hraničního uzlu, který předává informace
    - V každé AS ( nebo také někdy area ) může být použito jiné směrová
    - Směrování mezi AS zajišťují EGP protokoly - například BGP



zdroj: <https://bethepacketsite.wordpress.com/2016/04/20/dynamic-routing-path-vector/>

# Kategorizace směrování: Dle metod rozhodování: Distribuované: EGP: Path Vector: BGP

- BGP – Border gateway protokol
- Nejpoužívanější představit EGP
- Používá Path-vector
- Podporuje autentizaci
- Komunikuje pomocí 179/TCP
- Na rozdíl od IGP protokolů BGP nehledá sousedy, ale má je zadané
  - Bavíme se o propojení AS/ISP, tedy typicky point-to-point spoje – hledání nemá smysl
- Používá hledání nejkratší cesty, kde nejkratší je ta, která prochází přes co nejméně AS
  - Ceny jednotlivých cest je také možné doplnit o statické části ceny – konstanty
    - To je pro EGP nesmírně důležité, protože se zde typicky řeší násobné spoje a s ohledem ceny je nutné mít možnost preference
- Při navázání spojení se sousedem jsou nejprve vyměněny všechny směrovací
- V dalším kroku už se posílají jen změny – šetříme přenos
- Cyklicky se kontroluje, zda všichni sousedi žijí
  - Typicky 1x za minutu



zdroj: <https://www.semanticscholar.org/paper/BIGP-a-new-single-protocol-that-can-work-as-an-igp-Gupta/20bd5da8b6a4dfa2ea862f096953e2ce2d9a373b>

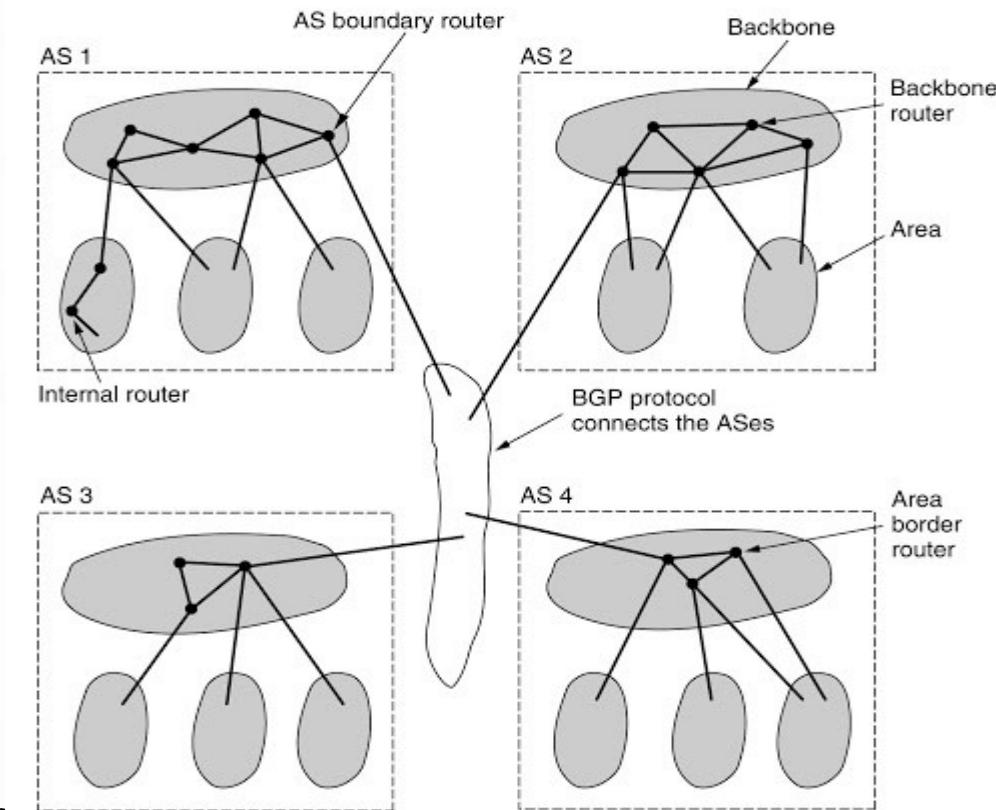
# Kategorizace směrování: Dle metod rozhodování: Distribuované: EGP: Path Vector: BGP: Metrika

- Metrika u BGP není tak jednoduchá / jednoznačná jako u IGP protokolů, kde hledáme jen nejlepší cestu v rámci jednoho organizačního celku
- BGP směruje provoz mezi ISP / či přes ně a v tu chvíli hrají roli i další faktory:
  - Ceny linek / Ceny přenosů
  - Obchodní podmínky / vztahy jednotlivých ISP
- Parametrů je více – viz tabulka a mají různou váhu a dosah
  - Část parametrů zůstává jen v AS a k dalším ISP se nepřenáší
  - Část parametrů se exportuje jako preference i mimo AS
    - Weight – jen v routeru – lokální
    - Local\_preference – v rámci AS
    - Med – exportované mezi AS

Priority	Attribute
1	Weight
2	Local Preference
3	Originate
4	AS path length
5	Origin code
6	MED
7	eBGP path over iBGP path
8	Shortest IGP path to BGP next hop
9	Oldest path
10	Router ID
11	Neighbor IP address

# Kategorizace směrování: Dle metod rozhodování: Hierarchické směrování

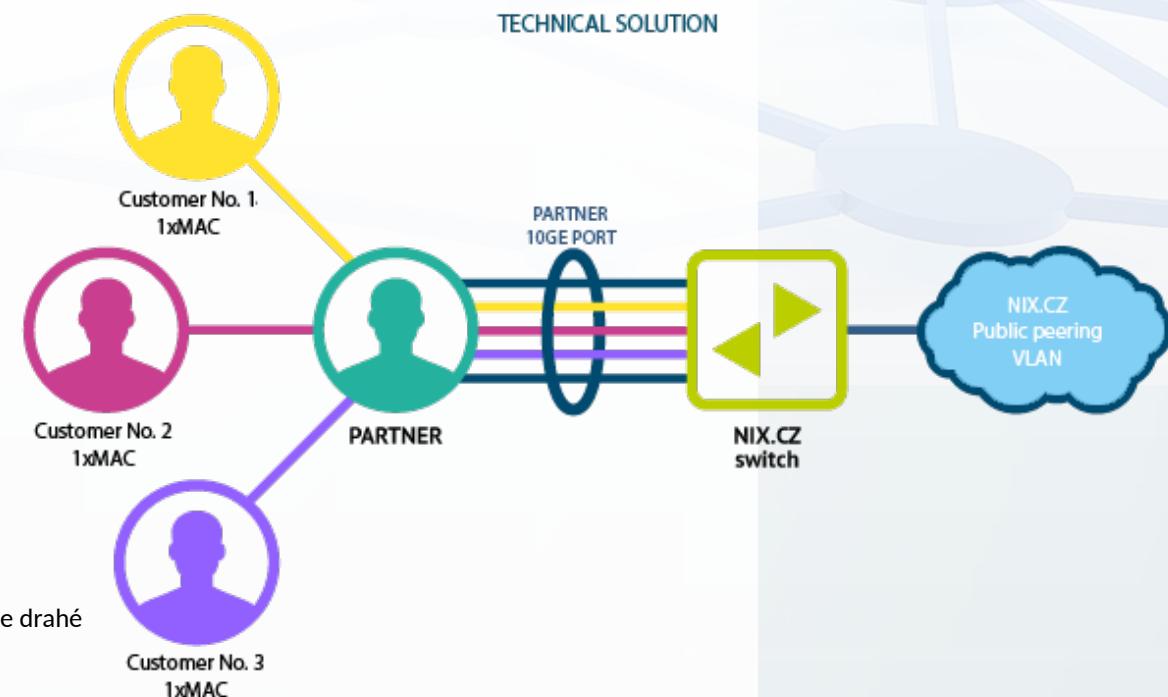
- Hierarchické směrování se snaží řešit problematiku rozsáhlých sítí
  - V rozsáhlých sítích vzniká problém s rychlosťí konvergence
    - Příliš dlouho trvá, než se nová informace dostane všude
  - Vzniká problém zatížení sítě obslužným provozem
    - Čím více routeru v jedné síti, tím více komunikace mezi nimi
- Řešením je rozdělit síť na více malých oblastí v a směrování řešit samostatně
  - Uvnitř dané oblasti
    - Nazývané směrovací doména / Area / Autonomní systém
    - Protokoly IGP
  - Externě mezi jednotlivými oblastmi
    - Protokoly EGP
- Historicky musely jednotlivé oblasti opravdu tvořit hierarchii
  - Počítalo se s páteřní oblastí na kterou se ostatní připojí – jako např v OS
  - Dnes už to není nutné a jednotlivé AS mohou být na stejně úrovni



zdroj: <https://lh3.googleusercontent.com/proxy/fWaDz-xR3WchuzrSYqH6xNal49OxF15mYi7-sX6k98b2YFbAr3sdLjtQhQkG3XnkJK-NL4VIjlenLtgcnNwQM1Ae0y1WlvLMV50yopELdGIqLDvR8co>

# Kategorizace směrování: Dle metod rozhodování: Hierarchické směrování: Peering

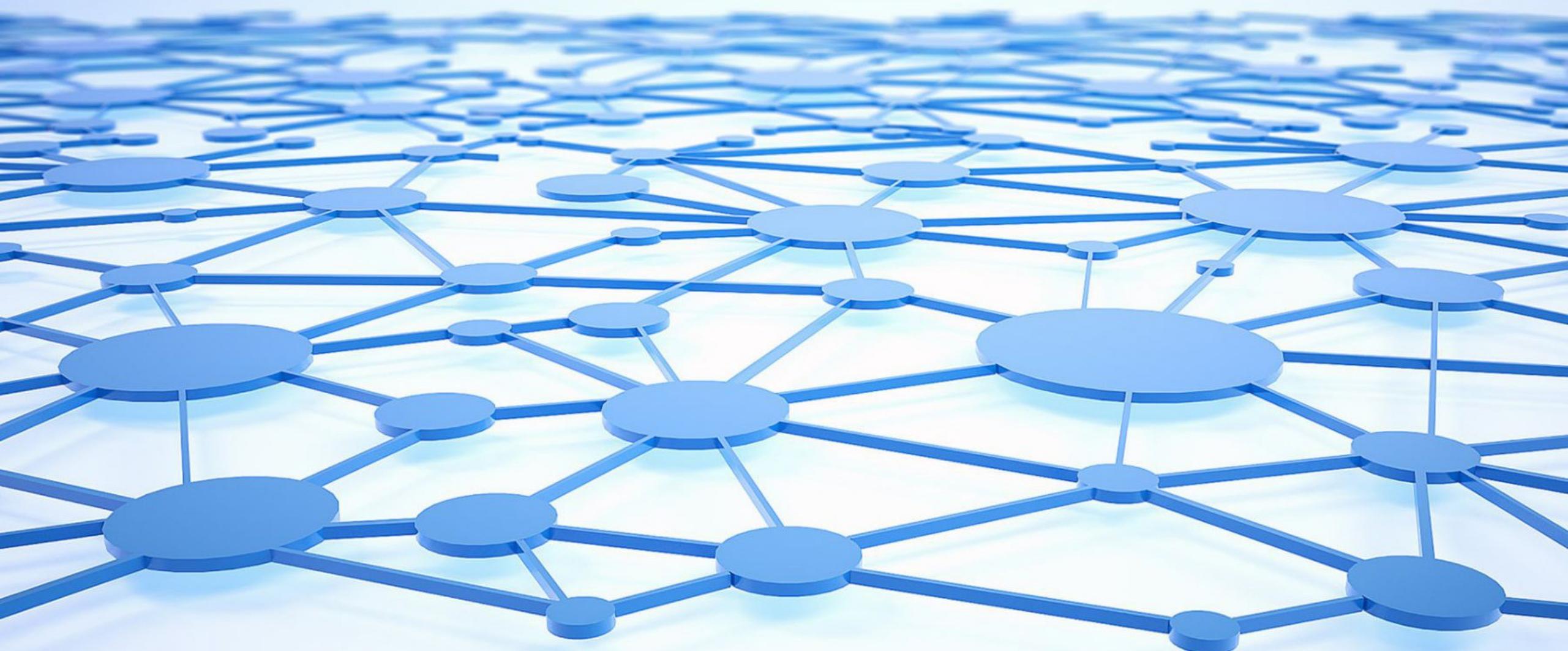
- Peering je vzájemné propojení jednotlivých ISP
  - Tedy propojení jednotlivých AS
- Peering je možné realizovat dvojím způsobem
  - Na přímo
    - Nejjednodušší cesta – dva ISP se dohodnou, udělají mezi sebou propoj a nastaví směrování kde si mezi sebou propagují informace o svých sítích
    - Problém je, že nemůžeme propojit na přímo všechny sítě
  - Pomocí peering centra
    - Více ISP se dohodne na spolupráce a společném „bodě“ kde bude docházet ke společnému propojení – peerigový uzel
    - Peeringové centrum je typicky hrazeno z členských poplatků
      - Je zde společný zájem na fungování
      - Fungování není levné protože zde tečou obrovská data
      - Jedná se citlivý bod z hlediska bezpečnosti
    - Cena za členství není malá a jednotlivý partneři se mohou propojovat vzájemně na přímo
      - Ať už jako primární nebo jako záložní řešení
- Propojení je možné na více úrovních, u nás se typicky řeší čtyři možnosti
  - NIX – české peeringové centrum
  - SIX – propojení na slovenská peeringové centrum
  - Transit – zahraniční konektivita
  - Přímé propojení ISP
- Ceny za jednotlivé přenosy a stejně tak parametry linek v jednotlivých propojích jsou různé a je nutné je hlídat
  - Protože pokud například tečou dlouhodobě data k českému partnerovi pomocí zahraniční konektivity bude to velice drahé



# Úvod do počítačových sítí

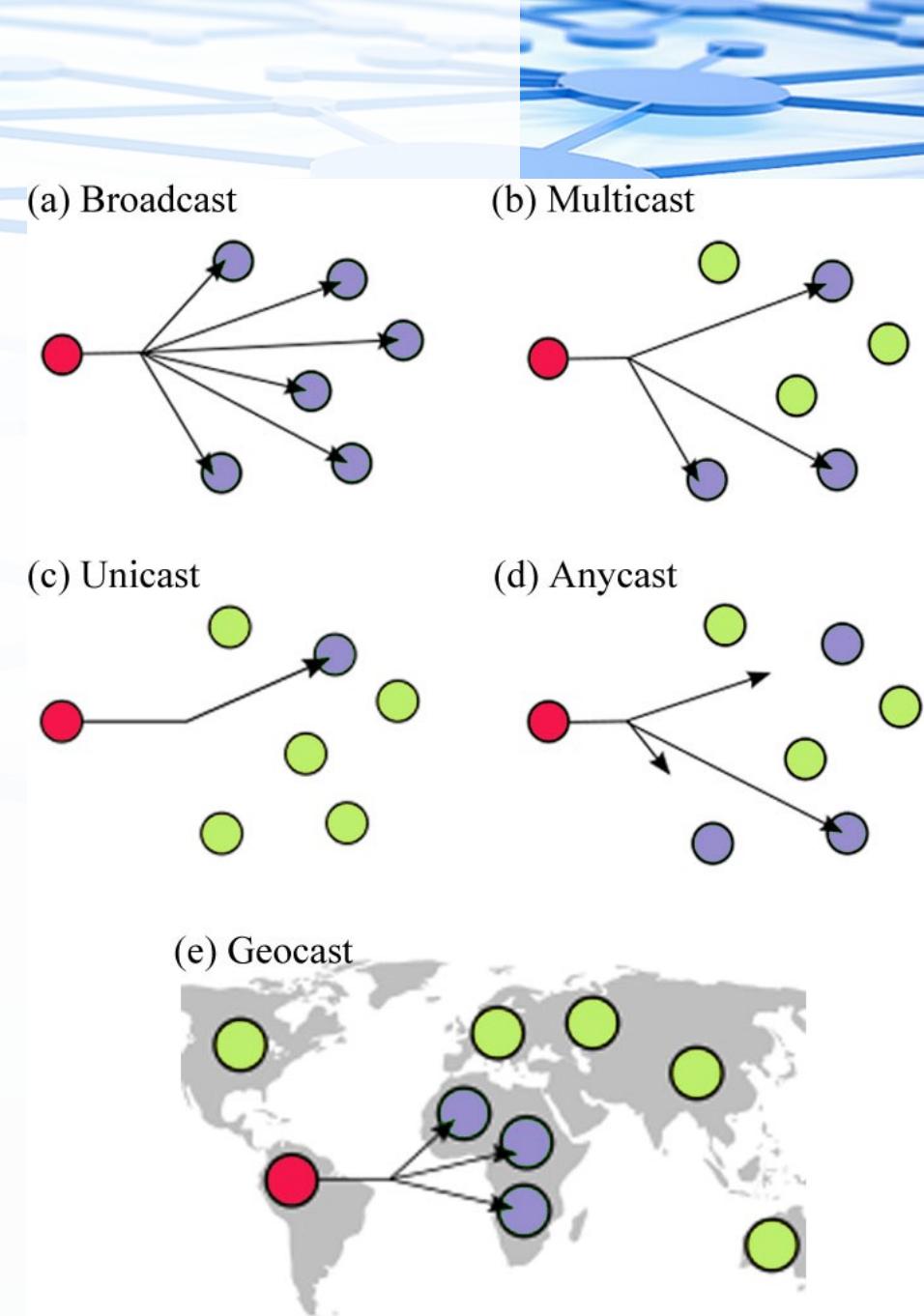
Přednáška 9 ( 2025/2026 )

ver. 2025-11-04-01



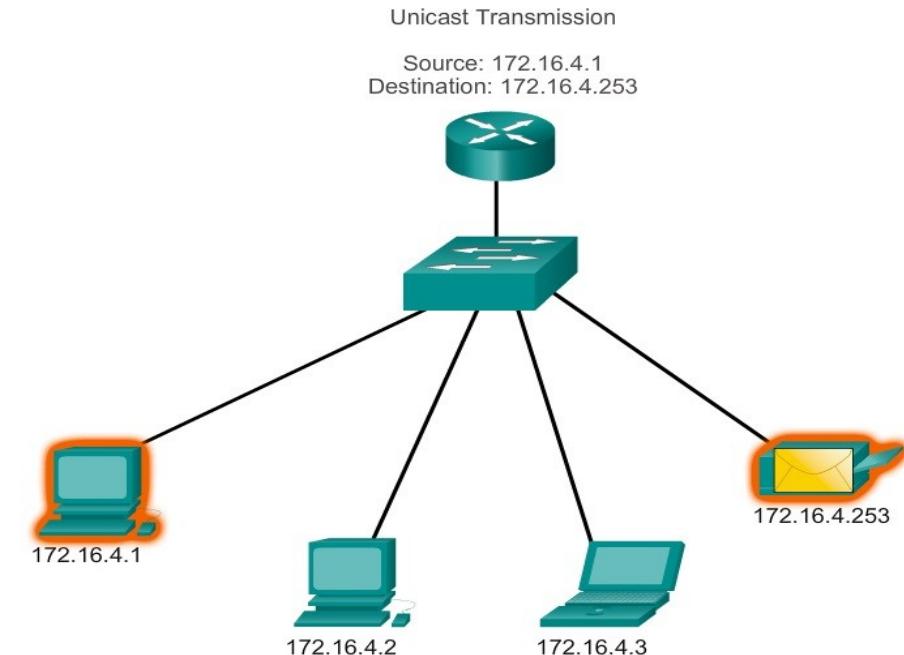
# L3 – Sítová vrstva II.

- Typy přenosů na L3
  - Unicast
    - Samostatné vysílání - „jeden s jedním“
  - Broadcast
    - Všesměrové vysílání - „jeden všem v dané síti“
  - Multicast
    - Skupinové vysílání - „jeden všem v dané skupině“
  - Anycast
    - Skupinové vysílání nejbližšímu členové - „jeden nejbližšímu členové dané skupiny“
  - Geocast
    - Skupinová vysílání dané lokaci - „jeden všem v dané geolokaci“

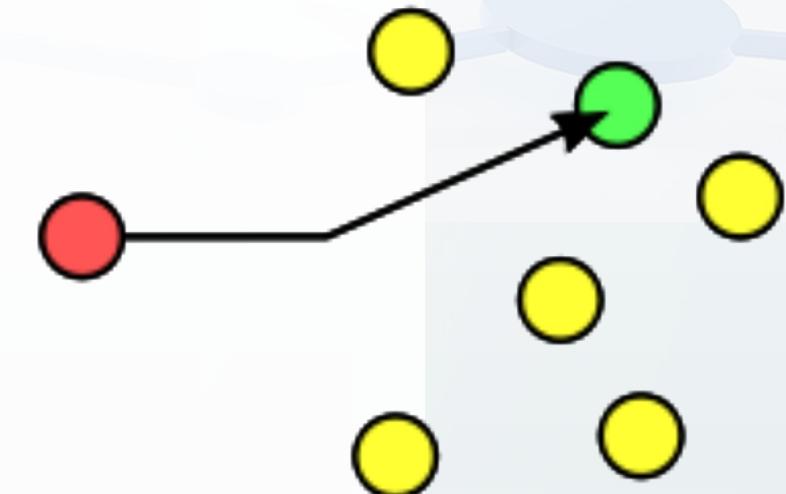


# Unicast

- Nejběžnější forma přenosu
- V zásadě se dá říci, že se jedná o přenos 1:1
  - Jeden účastník jednomu jinému účastníkovi
- Přenos může být realizován v rámci stejné LAN
  - Bez použití L3 směrování
- Přenos může být realizován mezi více LAN či WAN
  - S použitím L3 směrování
- Typické použití
  - Téměř veškeré běžně používané služby
    - HTTP, SMTP, POP/IMAP, FTP ....
- Nenáročný z pohledu zatížení sítě



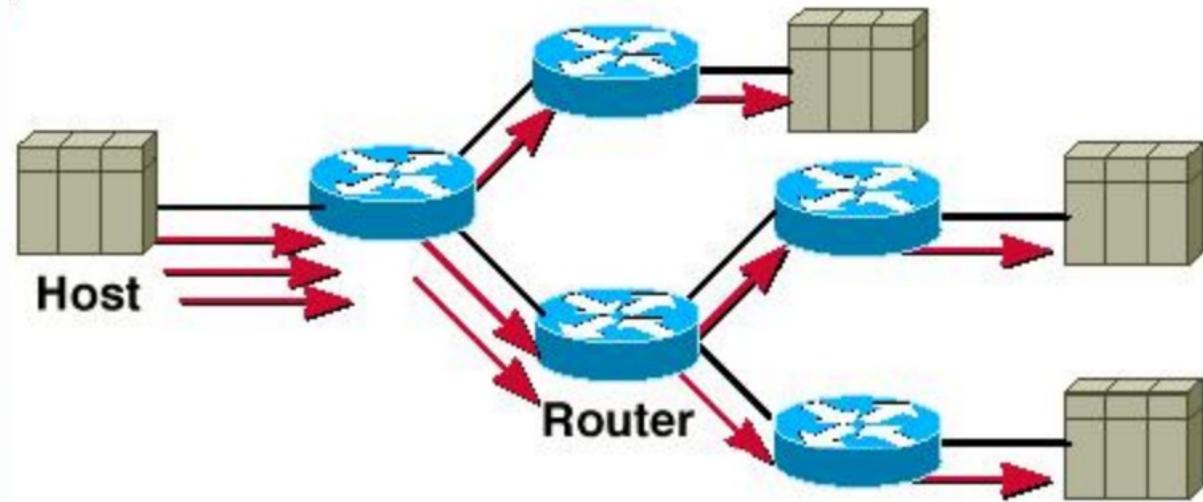
zdroj: <https://www.hitechmv.com/ipv4-unicast-broadcast-and-multicast/>



zdroj: <https://cs.wikipedia.org/wiki/Unicastarticle.asp?p=2180210&seqNum=12>

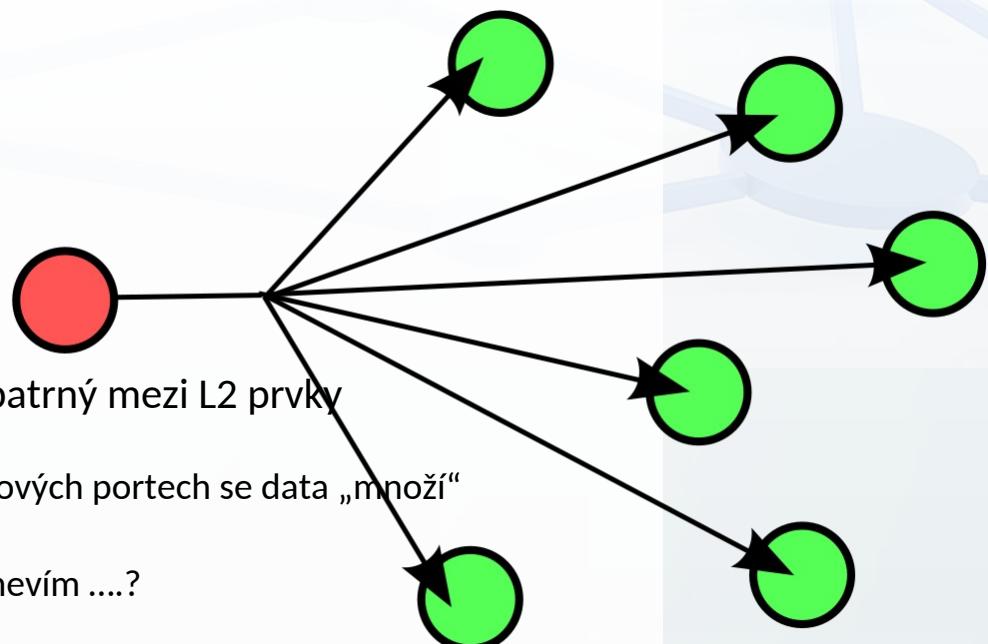
# Replikovaný Unicast

- Pokud potřebujeme odesílat data více účastníkům, musíme v případě použití unicstu přenosy opakovat
- Pro každého účastníka jsou ze zdroje odeslána vlastní data
- Výhodou je, že jednotlivé „streamy“ na sebe nejsou vázané
  - Tedy chyba v jednom nemusí ovlivnit více vysílání
    - Ale samozřejmě může, pokud je chyba ve zdroji data – např poškozený soubor
- Další výhodou je, že nepotřebuje žádné extra vybavení či adresaci
  - Proč také, jedná se o více běžných přenosů
- Nevýhod je v tomhle případě více
  - Vysílač musí znát všechny přijímače
    - Všichni k němu musí být připojeni
  - Data odchází tolíkrát, kolik přijímačů je připojeno
    - Což může být extrémně náročné na kapacitu linky a snadno může vést k saturaci linky a následně defakto k DDOS
      - Tolik účastníků chce čerpat data, až se zdroj stane nedostupný
- Reálně data neodchází ve stejný okamžik
  - Zde záleží na konkrétní službě – někde to může vadit někde nemusí
    - Pro sledování video prezentace to jistě nevadí
      - Sice to uvidím každý divák v drobném posunem, ale to ničemu nevadí
    - Pro sledování on-line prezentace s možností reagovat už to problém být může
      - Ptám se na něco co reálně proběhlo před „nějakým časem“ zpět
    - Hodně bude záležet na počtu klientů



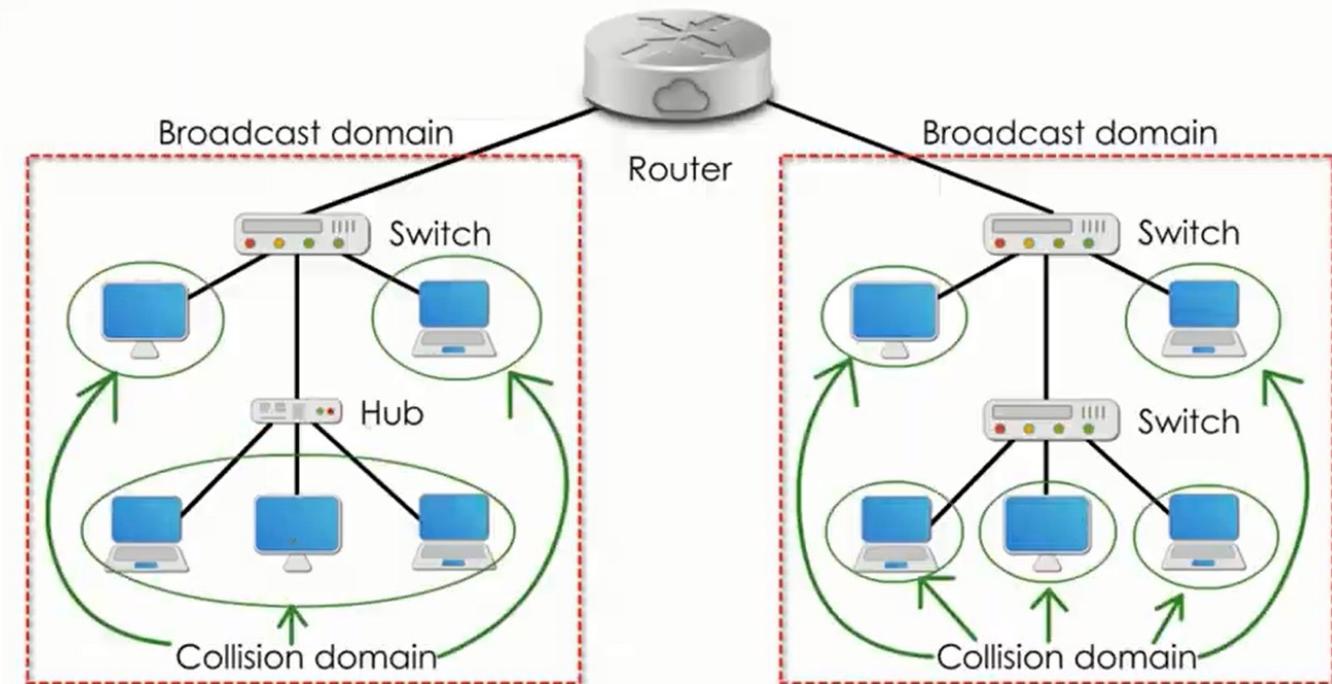
# Broadcast

- U unicstu nastane „neřešitelný“ problém pokud budu chtít oslovit všechna zařízení v síti
  - V uvozovkách proto, že v určitých krajních situacích si mohu pomoci výčtem
    - Např pokud sem součástí sítě 192.168.1.0/29, která reálně může obsahovat stroje 192.168.1.1-6 mohu obeslat všechny, ale pokud je v LAN více sítí, oslovím zas jen část
- Řešením je broadcast – tedy vysílání „jeden všem“
  - Všem == všem na které vidím
- Výhodou je, že se odesírají jen jedna data a to na speciální adresu, která identifikuje „všechny v dané L2 síti“
  - Nemusím tedy řešit kdo vše v síti je, jen „řeknu“ VŠEM
- Velice jednoduché na realizaci – díky speciální – broadcastové – adrese
- Nevýhodou je, že data musejí v síti přijmout a zpracovat všichni
  - Tedy i ti, kteří je nepotřebují
  - Dochází k zatěžování sítě i CPU na koncových stanicích
- Rozdíl mezi broadcastem a unicstrem pro více stanic z pohledu zatížení sítě, je patrný mezi L2 prvky
  - Unicast posílá kolik dat kolik je přijímačů – pro dva propojené switche
  - Broadcast, pro dva propojené switche, posílá data jen jednou a až na jednotlivých koncových portech se data „množí“
- Využití má všude tam, kde potřebuje oslovit všechny ve stejné síti
  - DHCP – ještě nevím o síti nic a potřebuji se na nastavení zeptat, ale koho když o síti nic nevím ....?
  - ARP – nastavení sítě už znám, ale hledám převod IP x MAC
  - SMB – zjišťování okolních PC ve stejné síti



# Broadcast – Broadcastová doména

- Pokud se tedy broadcast šíří všude, co jej zarazí ?
- L3 prvek - směrovač/router
- Broadcastová doména je oblast, kde se šíří broadcast a je limitovaná směrovačem
- Kolizní doména – jak už víme – je oblast, kde může dojít ke kolizi – souběžnému a neoddělitelnému vysílání
  - Kolizní doména je
    - Jeden port switche / bridge / routeru
    - Všechny porty hubu



# Broadcast – typy broadcastu

- Broadcast můžeme řešit na více úrovních ISO/OSI
  - L2 a L3
- Podle použité úrovně se budou odesílat různá data na různé adresy
  - Na L2 rámce na L3 pakety
- Zároveň se bude lišit dosah broadcastové komunikace
  - Tedy kam až se může vysílaná zpráva šířit
- Rozlišuje tři typy broadcastových zprav:
  - L2 Broadcast
  - Lokální L3 broadcast
  - Cílený L3 broadcast

# Broadcast – typy broadcastu: L2 broadcast

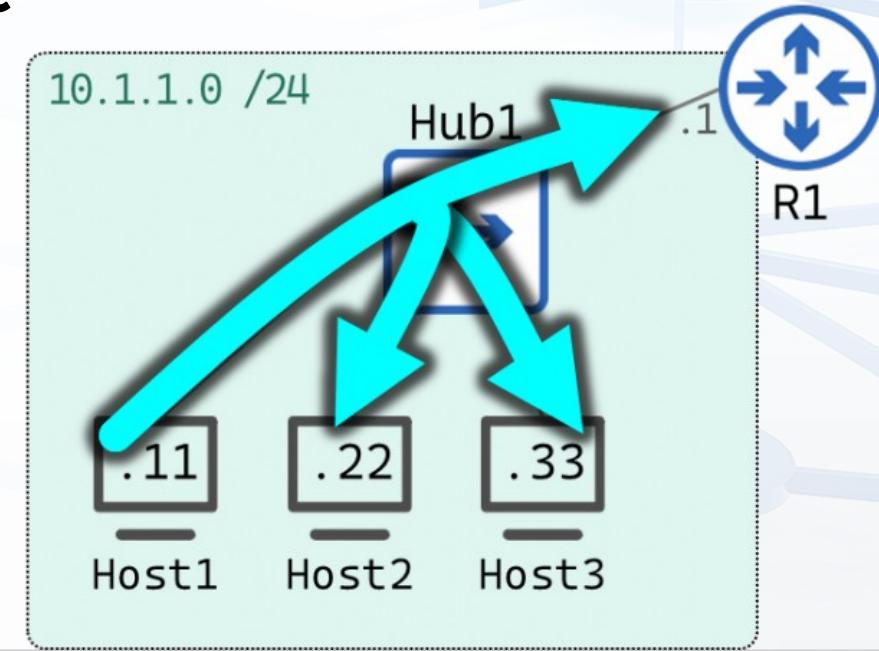
- Základní varianta všesměrového vysílání je realizována na L2 ISO/OSI
- Posílaná data mají charakter rámce / framu
- To, že se jedná o broadcast je definovanou adresou příjemce
- Pro Ethernet je MAC pro L2 broadcast FF:FF:FF:FF:FF:FF
  - Samé jedničky
- Takový rámec je
  - Na hubu kopírován na všechny porty kromě příchozího
    - Hub to ani jinak neumí – jedná se o více-portový opakovač
  - Switch / bridge jej kopírují na „všechny“ porty kromě příchozího
    - Kopírují rámec – ne jen signál
    - Na „všechny“ porty – protože switch může obsahovat VLAN
    - Na vše krom příchozího aby nedošlo k zacyklení – broadcastové bouři
      - Ono k ní stejně může dojít pokud je v síti smyčka, což lze řešit pomocí STP protokolu
  - Router na takový rámec může odpovět, ale **dále jej nešíří**
    - Rozuměj na další porty

Protocol	Source	Destination	Info
ICMP	10.1.1.11	255.255.255.255	Echo (ping) request id=0x6801, seq=0/0, ttl=64 (broadcast)
ICMP	10.1.1.33	10.1.1.11	Echo (ping) reply id=0x6801, seq=0/0, ttl=64
ICMP	10.1.1.22	10.1.1.11	Echo (ping) reply id=0x6801, seq=0/0, ttl=64
ICMP	10.1.1.1	10.1.1.11	Echo (ping) reply id=0x6801, seq=0/0, ttl=255

> Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
> Ethernet II, Src: ee:ee:ee:11:11:11, Dst: ff:ff:ff:ff:ff:ff  
> Internet Protocol Version 4, Src: 10.1.1.11, Dst: 255.255.255.255  
> Internet Control Message Protocol

# Broadcast – typy broadcastu: Lokální L3 broadcast

- Lokální nebo také místní či běžný broadcast
- Posíláme L3 paket a posíláme jej na L3 broadcastovou adresu
  - Např pro Ipv4 je to 255.255.255.255
  - Stejně jako na L2 i na L3 se jedná o adresu, kde jsou binárně samé „1“
- Realizace bude provedena tak, že se tento paket umístí do L2 rámce s adresou FF:FF:FF:FF:FF a odešle se jako L2 broadcastový rámec
- Odpovědi docházejí už na konkrétní L3 adresu odesílatele
- Hub / Switch / Bridge danou zprávu posírají dále
- Router opět jen může odpovídět, ale na další porty ji nekopíruje
  - Protože ví, že se jedná o broadcast
- Stejně jako u L2 je možné realizovat jen v rámci jedné LAN
  - Logicky, protože router nás jinam nepustí

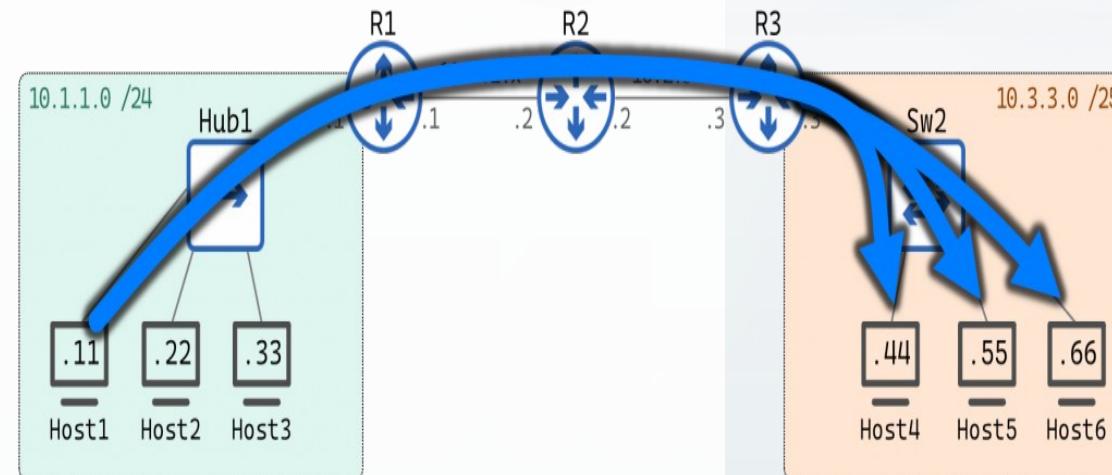


Protocol	Source	Destination	Info
ICMP	10.1.1.11	255.255.255.255	Echo (ping) request id=0x6801, seq=0/0, ttl=64 (broadcast)
ICMP	10.1.1.33	10.1.1.11	Echo (ping) reply id=0x6801, seq=0/0, ttl=64
ICMP	10.1.1.22	10.1.1.11	Echo (ping) reply id=0x6801, seq=0/0, ttl=64
ICMP	10.1.1.1	10.1.1.11	Echo (ping) reply id=0x6801, seq=0/0, ttl=255

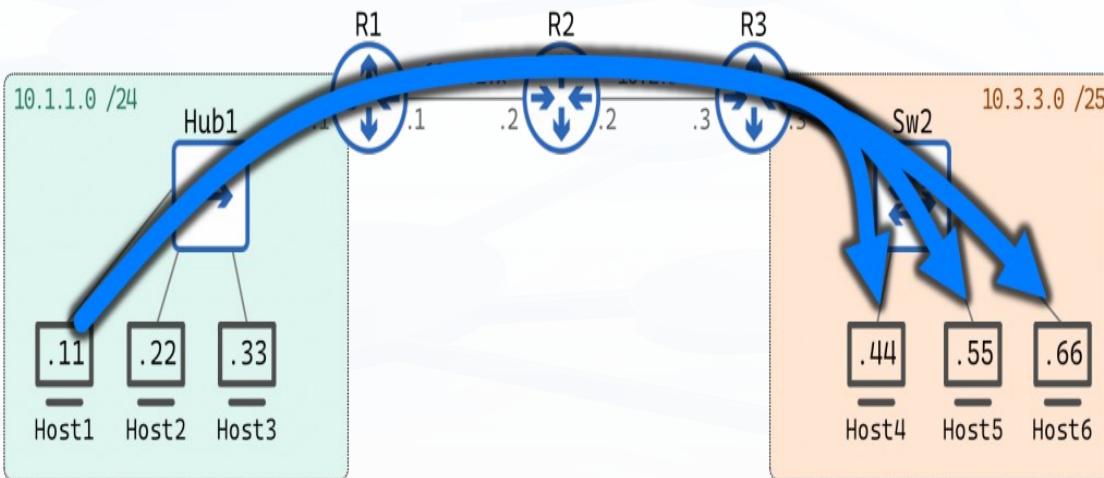
> Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
> Ethernet II, Src: ee:ee:ee:11:11:11, Dst: ff:ff:ff:ff:ff:ff  
> Internet Protocol Version 4, Src: 10.1.1.11, Dst: 255.255.255.255  
> Internet Control Message Protocol

# Broadcast – typy broadcastu: Cílený L3 broadcast

- Podobě jako v předchozím případě u lokálního broadcastu posíláme paket, ale zde na jinou adresu
- Už nepoužíváme obecnou adresu 255.255.255.255, ale používáme broadcastovou adresu sítě, do které chceme vysílat
  - Např u IPv4 se jedná o nejvyšší adresu v síti – síť je jak víme definovaná maskou
    - Ona může být definovaná i třídou adres pro classfull sítě, ale třídy adresy pojí s konkrétní výchozí maskou
      - A 8, B/16, C/24
  - Např pro 10.1.1.0/24 bude broadcastová adresa 10.1.1.255
    - Tedy binárně opět samé „1“, ale JEN ve variabilní části adresy
- Pokud se je jedná o síť, jejíž jsem součástí, je situace stejně jako u lokální broadcastu
- Ale pokud se nejedná o lokální síť je situace dramaticky odlišná
  - Tento paket je v počátku šířen jako unicast
    - Logicky, protože pro 10.1.1.0/24 je 10.1.1.255 broadcast, ale pro 10.1.0.0/16 je to plnohodnotná adresa
  - Tento paket tedy projde routerem/routery až do cílové sítě
    - Na základě běžného směrování
  - A teprve až na routeru, který spravuje cílovou síť, začne být šířen jako lokální broadcast
  - Protože cílový router má na jednom interface síť 10.1.1.0/24 a tedy ví, že se jedná o broadcastovou adresu
  - L2 broadcast se použije až na posledním routeru
- Odpověď jde pak klasickým unicast paketem
  - A řídí se klasickým směrováním
- Povolení tohoto typu komunikace představuje bezpečnostní riziko
  - Mohl posílat požadavky do cizích sítí a zjišťovat živé stroje
  - Typicky se defaultně zakazuje



# Broadcast – typy broadcastu: Cílený L3 broadcast - příklad



Protocol	Source	Destination	Info
ICMP	10.1.1.11	10.3.3.127	Echo (ping) request id=0x6b01, seq=0/0, ttl=64 (no response f...
ICMP	10.2.3.3	10.1.1.11	Echo (ping) reply id=0x6b01, seq=0/0, ttl=253
ICMP	10.3.3.66	10.1.1.11	Echo (ping) reply id=0x6b01, seq=0/0, ttl=61
ICMP	10.3.3.55	10.1.1.11	Echo (ping) reply id=0x6b01, seq=0/0, ttl=61
ICMP	10.3.3.44	10.1.1.11	Echo (ping) reply id=0x6b01, seq=0/0, ttl=61

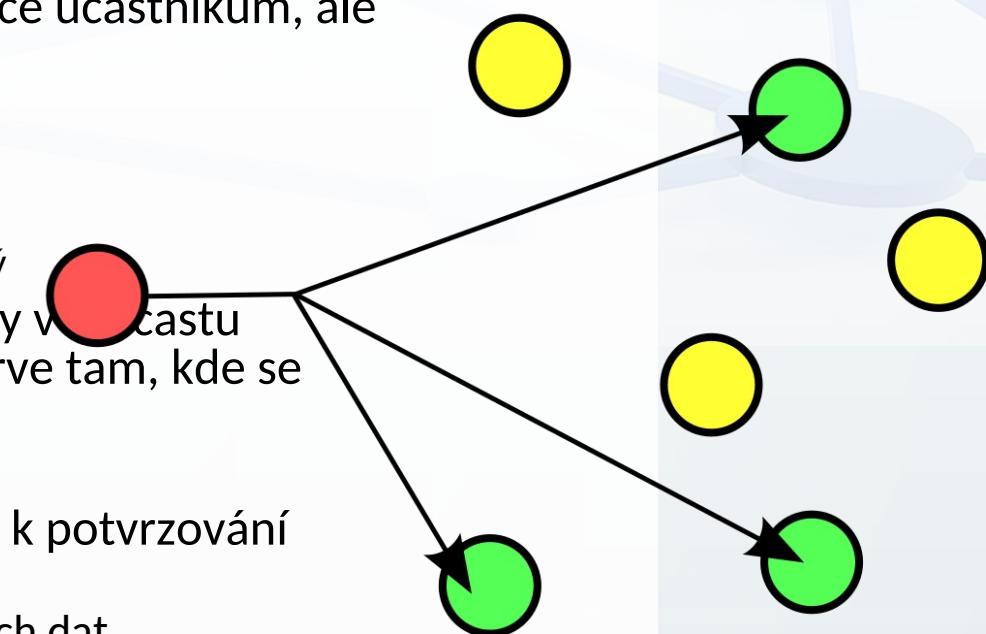
> Frame 13: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
> Ethernet II, Src: ee:ee:ee:11:11:11, Dst: ee:ee:10:11:11:11  
> Internet Protocol Version 4, Src: 10.1.1.11, Dst: 10.3.3.127  
> Internet Control Message Protocol

Protocol	Source	Destination	Info
ICMP	10.1.1.11	255.255.255.255	Echo (ping) request id=0x6b01, seq=0/0, ttl=61 (broadcast)
ICMP	10.3.3.66	10.1.1.11	Echo (ping) reply id=0x6b01, seq=0/0, ttl=64
ICMP	10.3.3.55	10.1.1.11	Echo (ping) reply id=0x6b01, seq=0/0, ttl=64
ICMP	10.3.3.44	10.1.1.11	Echo (ping) reply id=0x6b01, seq=0/0, ttl=64

> Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
> Ethernet II, Src: ee:ee:10:33:33:33, Dst: ff:ff:ff:ff:ff:ff  
> Internet Protocol Version 4, Src: 10.1.1.11, Dst: 255.255.255.255  
> Internet Control Message Protocol

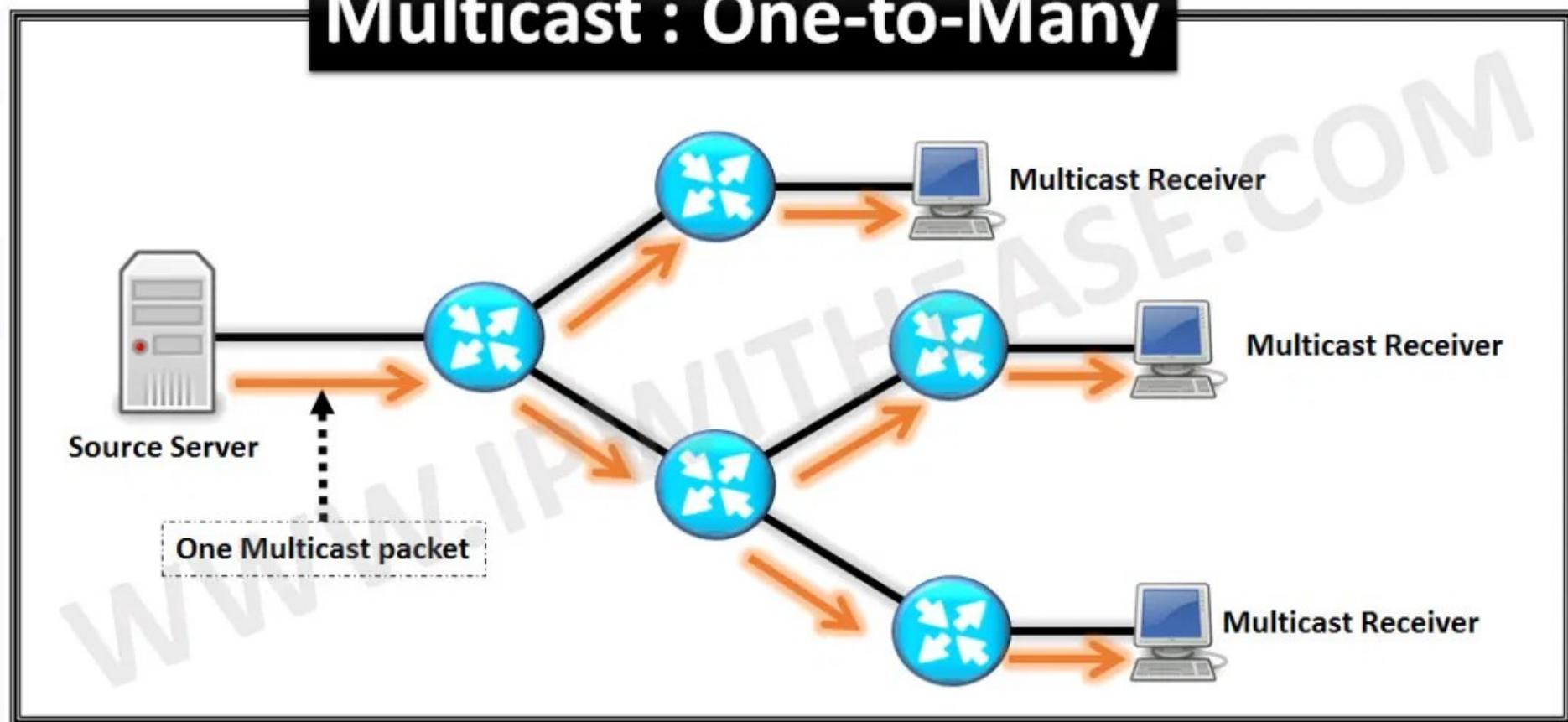
# Multicast

- Broadcastová komunikace umí poslat data všem v rámci jedné LAN
  - Ale data jdou VŠEM v dané LAN
- Replikovaný unicast umí poslat data na více příjemců i v různých sítích
  - Ale za cenu tolika datových toků od zdroje kolik je příjemců přenosu
- Multicast je řešení pro situace, kdy potřebuje poslat stejná data více účastníkům, ale
  - Nechceme / nemůžeme být omezeni na jednu LAN
  - Potřebujeme minimalizovat datové toky
    - To obecně chceme vždy ;)
  - Skupina příjemců není předem známa a ani nijak místně vázaná
    - Tohle neplatí tak obecně – musí být v rámci celé cesty multicast dostupný
- Multicast funguje tak, že tam kde jdou data společnou cestou, tedy v multicastu by bylo více paralelních přenosu, jsou přenášena jen jednou a teprve tam, kde se cesty dělí se kopírují na více portů
  - Ale jednou cestou / portem jde vždy jen jedna kopie data
- Multicast je nespolehlivý – aby spolehlivý byl, muselo by docházet k potvrzování a případně opakování dat – a to je problém pro „live“ data
  - Upřednostňujeme kontinuální datový tok, před jistotou doručení všech dat

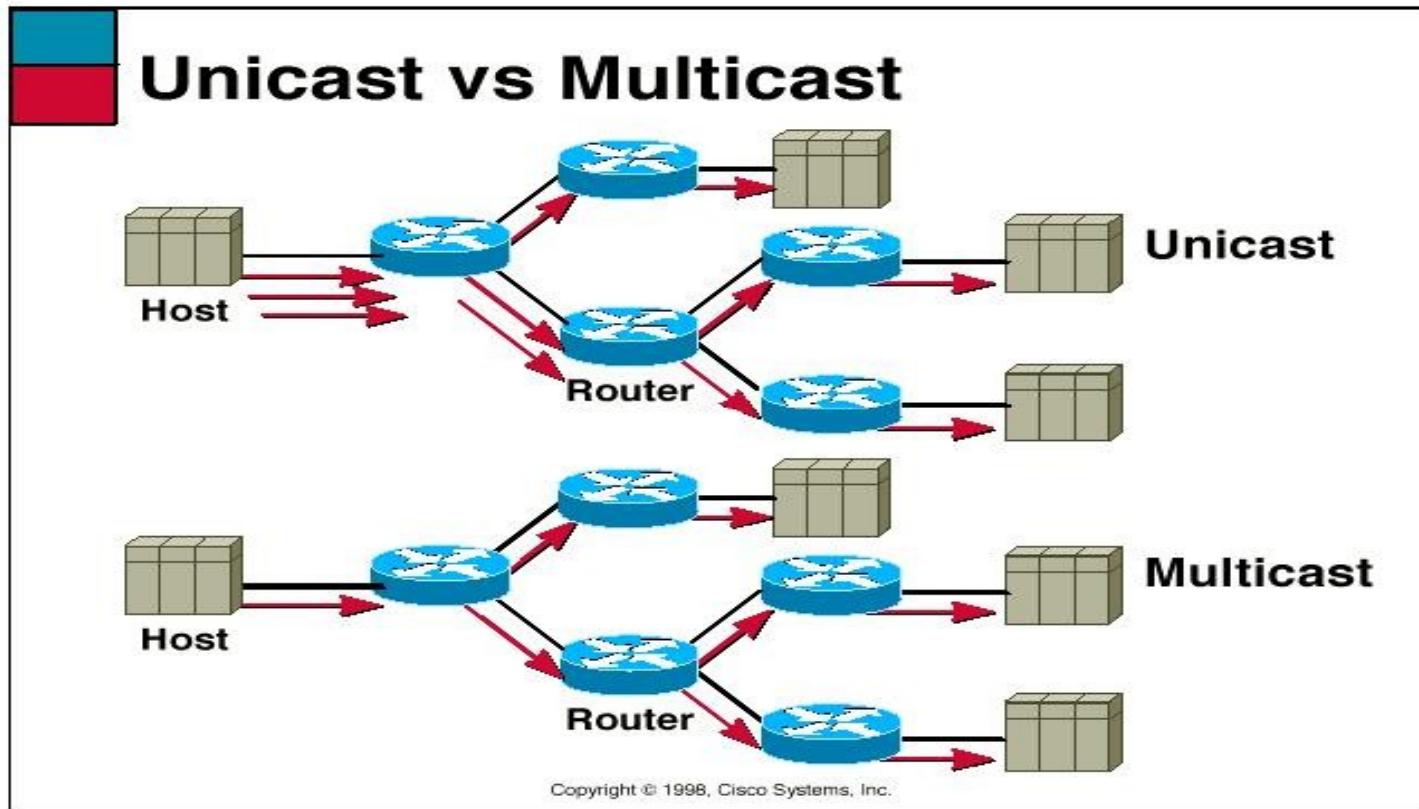


# Multicast - příklad

## Multicast : One-to-Many



# Multicast – porovnání s unicastem



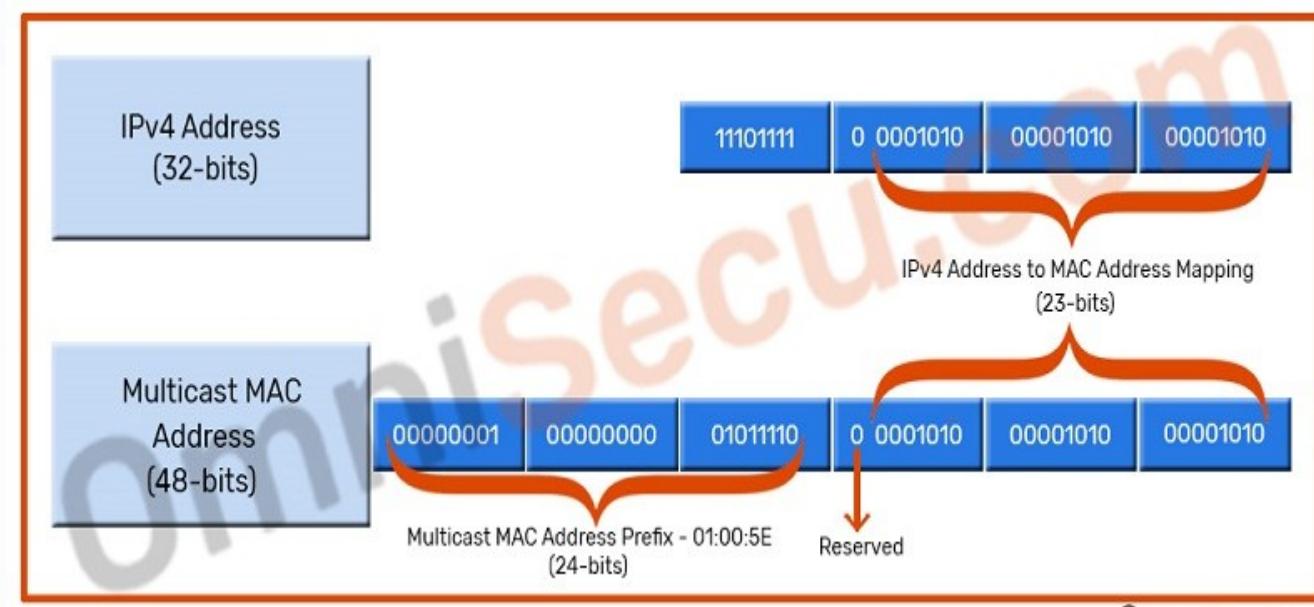
zdroj: <https://havel.mojeservery.cz/mikrotik-blokovani-omezovani-multicast-provozu/>

# Multicast - použití

- Typicky tam, kde potřebujeme distribuovat větší množství stejných dat ve stejný okamžik skupině příjemců
- Velmi často se používá pro distribuci video/zvukového vysílání
  - Stream videa
  - Videokonference
- Distribuované interaktivní hry / simulace
  - Stejnou situaci potřebujeme aby vidělo naráz více účastníků
- Hromadné kopírování dat
  - Například klonování pevných disků
  - Na KIV používáme UDPCast ke klonování stanic
- Konfigurace skupin zařízení
  - Například NTP synchronizace času

# Multicast – Adresace

- Multicast, podobě jako broadcast se řeší na L2 i L3 úrovni
  - A tedy pro switch i router se chová zcela jinak
- Na úrovni L2 se používají speciální MAC adresy pro multicast
  - Ty tvoří pevně daný prefix
    - Pro Ethernet je to 01:00:5E
  - A následně 23 bitů přímo z L3 IP multicastové adresy
- Na úrovni L3 má multicast vyhrazené adresy třídy D
  - 224.0.0.0/4
  - 224.0.0.0 až 239.255.255.255
- Jednotlivé skupiny adres mají svůj předdefinovaný specifický význam



# Multicast – L3 adresy a jejich význam

IP multicast address range	Description	Routable
224.0.0.0 to 224.0.0.255	Local subnetwork <sup>[1]</sup>	No
224.0.1.0 to 224.0.1.255	Internetwork control	Yes
224.0.2.0 to 224.0.255.255	AD-HOC block 1 <sup>[2]</sup>	Yes
224.3.0.0 to 224.4.255.255	AD-HOC block 2 <sup>[3]</sup>	Yes
232.0.0.0 to 232.255.255.255	Source-specific multicast <sup>[1]</sup>	Yes
233.0.0.0 to 233.251.255.255	GLOP addressing <sup>[4]</sup>	Yes
233.252.0.0 to 233.255.255.255	AD-HOC block 3 <sup>[5]</sup>	Yes
234.0.0.0 to 234.255.255.255 <sup>[citation needed]</sup>	Unicast-prefix-based	Yes
239.0.0.0 to 239.255.255.255	Administratively scoped <sup>[1]</sup>	Yes

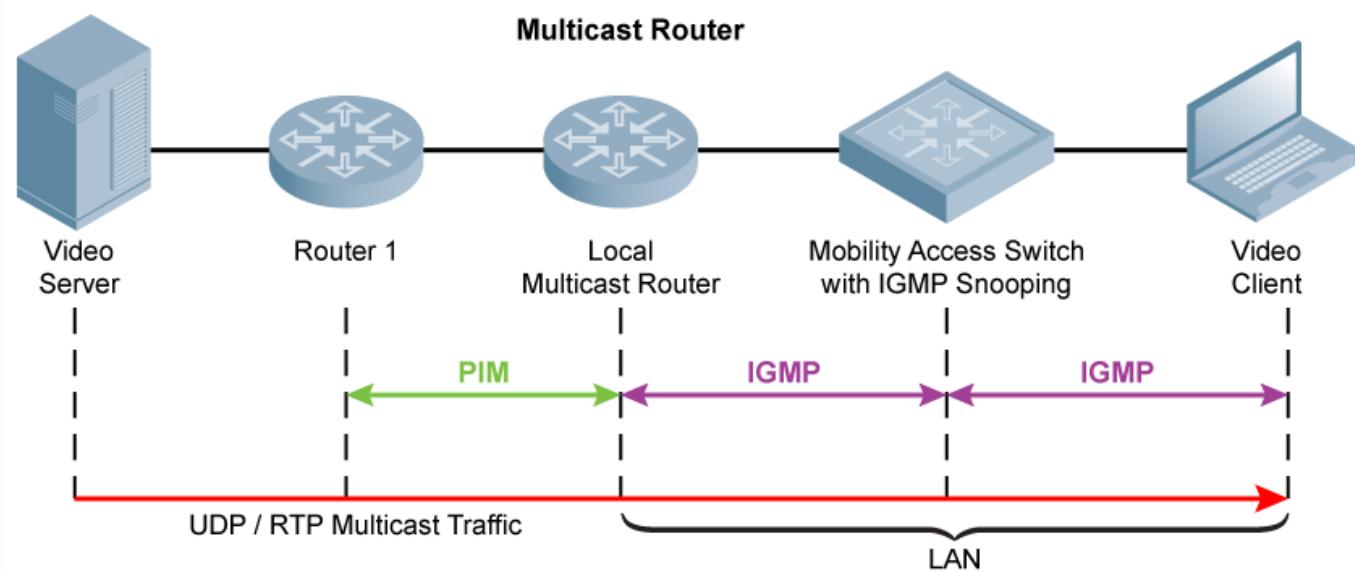
zdroj: [https://en.wikipedia.org/wiki/Multicast\\_address](https://en.wikipedia.org/wiki/Multicast_address)

Well-Known Reserved Multicast Addresses	
Address	Usage
224.0.0.1	All multicast hosts
224.0.0.2	All multicast routers
224.0.0.4	DVMRP routers
224.0.0.5	All OSPF routers
224.0.0.6	OSPF designated routers
224.0.0.9	RIPv2 routers
224.0.0.10	EIGRP routers
224.0.0.13	PIM routers
224.0.0.22	IGMPv3
224.0.0.25	RGMP

zdroj: <https://tutorzine.com/introduction-to-ip-multicasting/>

# Multicast – Princip fungování

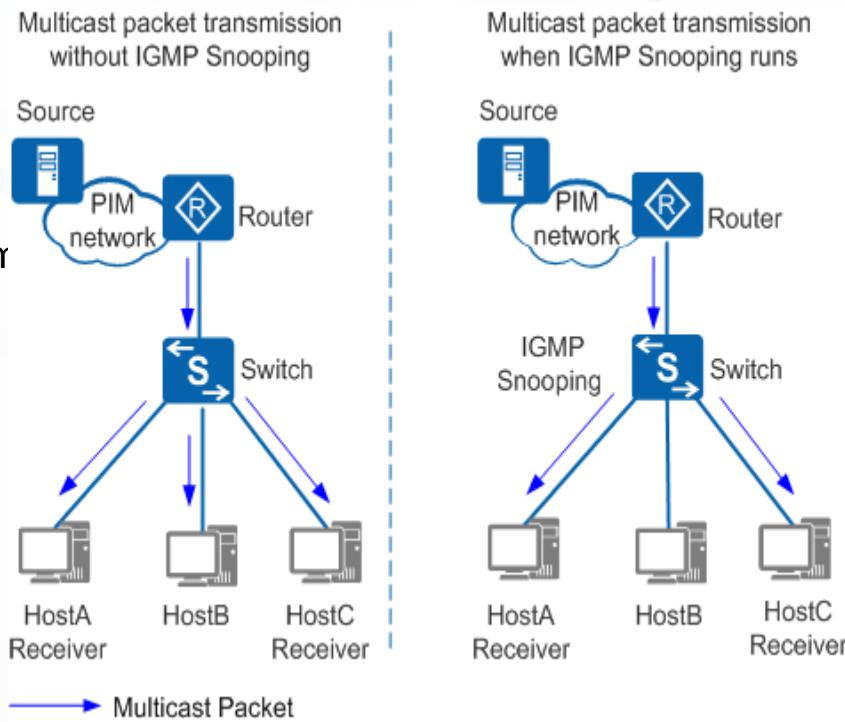
- V první řadě je nutné aby zařízení, která se mají multicastové komunikace účastnit ji podporovala a byla na nich povolena
  - S koncovými stanice obvykle problém není, ale na L2 i L3 prvcích se multicast často zakazuje
    - Důvodem je jak bezpečnost tak stabilita sítě, protože multicast může zařízení více zatěžovat
- Následně je nutné, aby se koncová stanice registrovala alespoň do jedné multicastové skupiny
  - Samozřejmě může i do více
  - Registrací stanice vyjadřuje přání přijímat data skupiny
  - K registraci s využívá protokol IGMP
- Samotný přenos se liší v rámci LAN a WAN
  - v L2 se použije multicastová MAC
  - V L3 je třeba použít multicastové směrování k doručování dat
    - Jedná se např o PIM-SM, PIM-DM, DVMRP, MOSPF



zdroj: <https://community.arubanetworks.com/blogs/arunhasan11/2020/10/20/how-to-configure-verify-and-troubleshoot-igmp-and-pim-sm-functionality>

# Multicast – Chování na L2 a L3

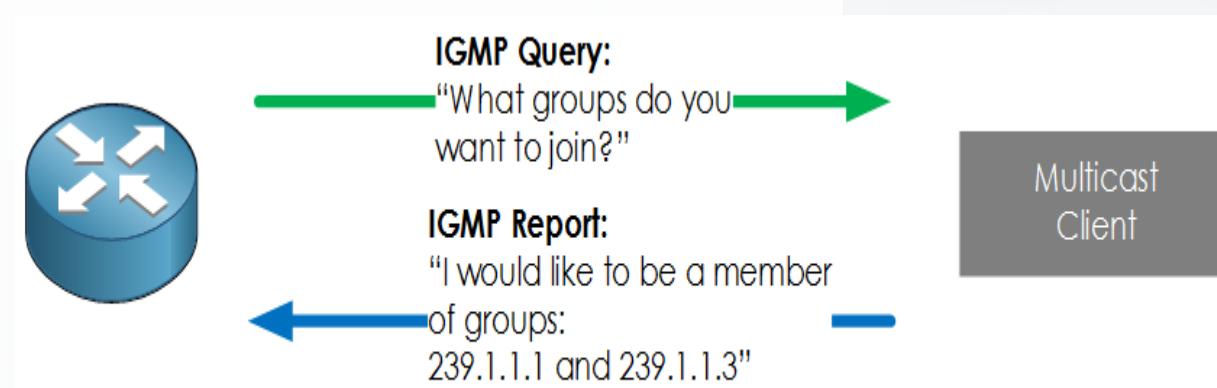
- Chování na L2
  - Velice podobné broadcast
  - Pro switch jsou dvě možné varianty
    - Bez IGMP Snoopingu – data se kopírují na všechny porty kromě příchozího
      - Opět je třeba řešit zacyklení - STP
    - S IGMP Snoopingem – data se kopírují jen na porty, kde je o daný multicast zájem
  - Na rozdíl od broadcastu se ale nezpracovává na všech stanicích, ale jen na těch, které mají o data zájem – jsou součástí skupiny
    - To poznají na základě MAC, ve které je část IP adresy
- Chování na L3
  - Zde je situace proti unicastu složitější, protože paket může být nutné zaslat na více portů
  - Je nutné předcházet zacyklení – proto se vytváří distribuční strom a data se posílají jen do větví
    - Kořenem stromu je buď první router u zdroje nebo například dohodnutý router



zdroj:  
<https://support.huawei.com/enterprise/en/doc/EDOC1100116611/a0355a52/igmp-snooping>

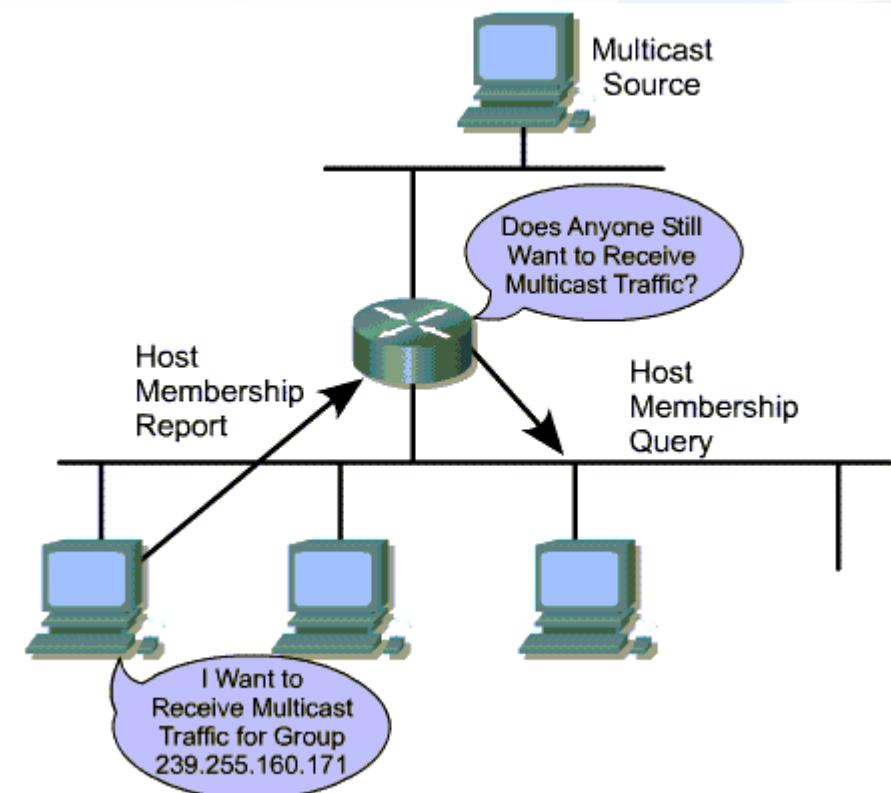
# Multicast – Registrace do skupiny: IGMP

- IGMP - Internet Group Management Protocol
- Slouží ke komunikaci mezi lokální stanicí a místním routerem
- Pokud stanice chce přijímat multicast protokol, musí se zaregistrovat na lokálním routeru
- Router musí vědět, že na daném portu – odkud přišel požadavek – je někdo kdo chce přijímat multicast data dané skupiny
- Zároveň routeru umožňuje zjišťovat, zda registrovaná zařízení mají o členství stále zájem
  - Aby se detekovaly mrtvé stanice /aplikace
  - A aby se tím snížil zbytečný trafik
  - Zjišťování probíhá opakovaně – protože se mění v čase
- Existují tři verze IGMP
  - IGMPv1
  - IGMPv2
  - IGMPv3



# Multicast – Registrace do skupiny: IGMPv1

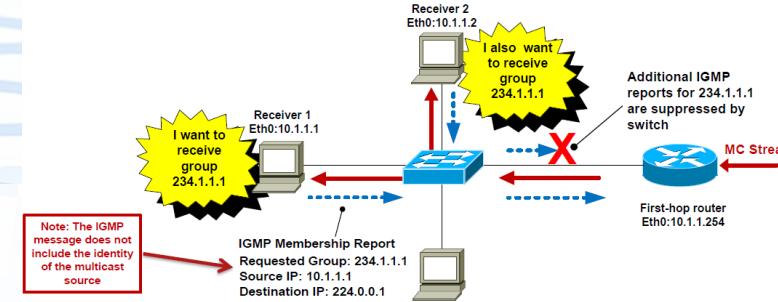
- Obsahuje dva typy zpráv
  - Reportovací
    - Klient informuje lokální router o tom, že chce být člen skupiny
  - Dotazovací
    - Router se ptá klientů, zda jsou stále ještě součástí skupin do kterých se
      - Cílem je eliminovat zbytečné přenosy
      - Pokud neodpoví nikdo, skupina na routeru zanikne
      - Dotaz se posílá každých 60s, s tím že timeout je 180s
      - Dotaz se posílá na 224.0.0.1 s TTL=1
        - » Neopustí lokální síť
      - Vzniká zde problém, že router se nedozví o tom, že stanice opustila skupinu
        - » Například proto, že klient ukončil aplikaci



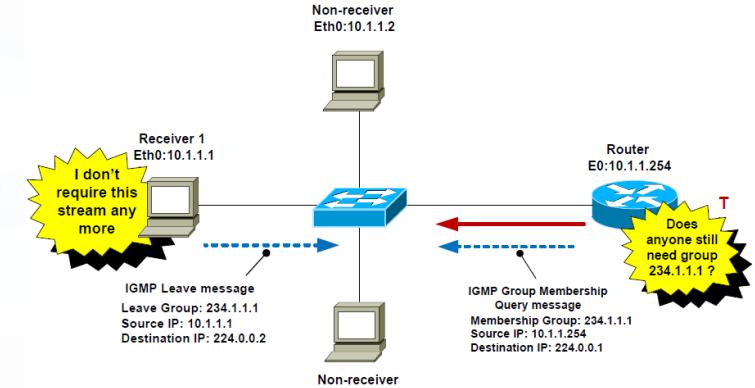
# Multicast – Registrace do skupiny: IGMPv2

- Vychází s IGMPv1 snaží se řešit jeho nedostatky
- Zavádí proti IGMPv1 čtyři novinky:
  - Zavádí Leave message, kde klient informuje server, že už není členem skupiny
    - Ta se posílá na IP 224.0.0.2
    - Urychluje ukončení zbytečných streamů a zároveň zánik prázdných skupin
  - Upravuje timeouty
    - Dotaz na hosty se posílá každých 125s
  - Zavádí volbu hlavního routeru v dané síti
    - Pokud je jich v jedné LAN více
    - Router s nejnižší IP ( porovnáno binárně )
    - Pokud zaslechnu dotazovací zprávu od routeru s nižší IP, přestane své zprávy na 400s posílat, pokud pak další neuslyší považuje původní stanici za mrtvou a začne posílat dotazy – čímž dojde k nové volbě
  - Zavádí specifická dotazy pro danou skupinu
    - Dokáže nově adresovat hosty v jednotlivých skupinách

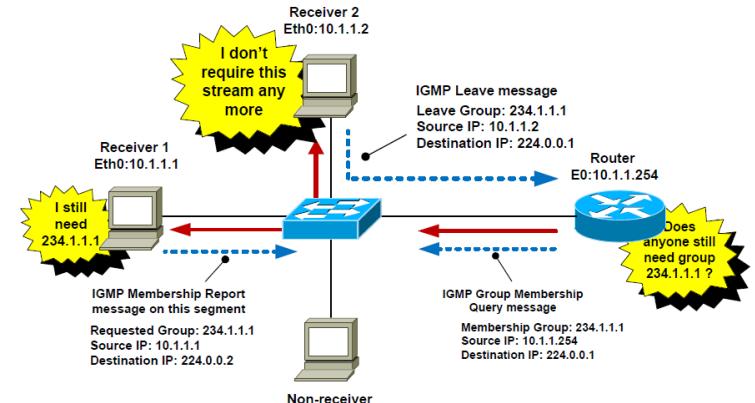
## IGMPv2 – Joining a Group



## IGMPv2 – Leaving a Group



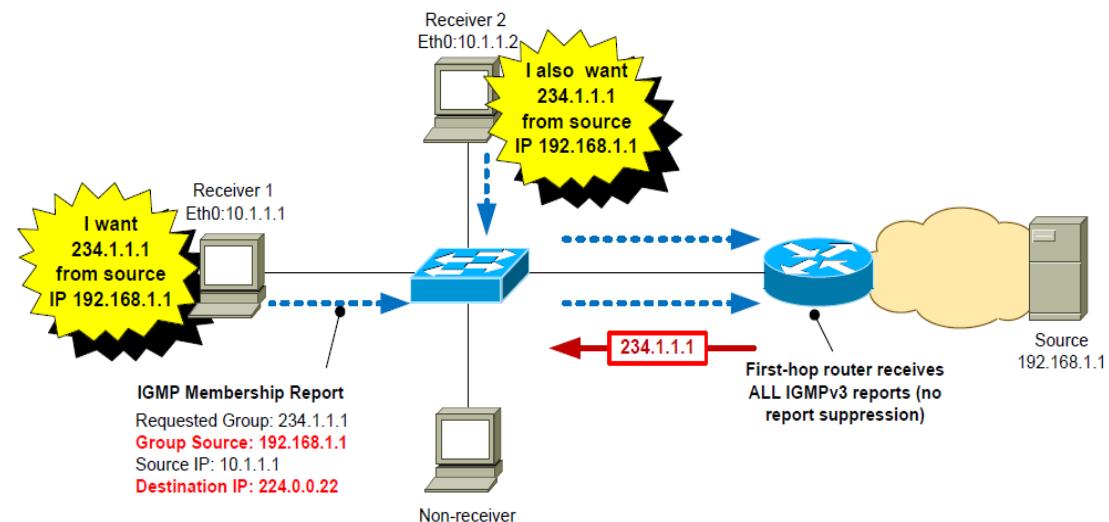
## IGMPv2 – Maintaining a Group



# Multicast – Registrace do skupiny: IGMPv3

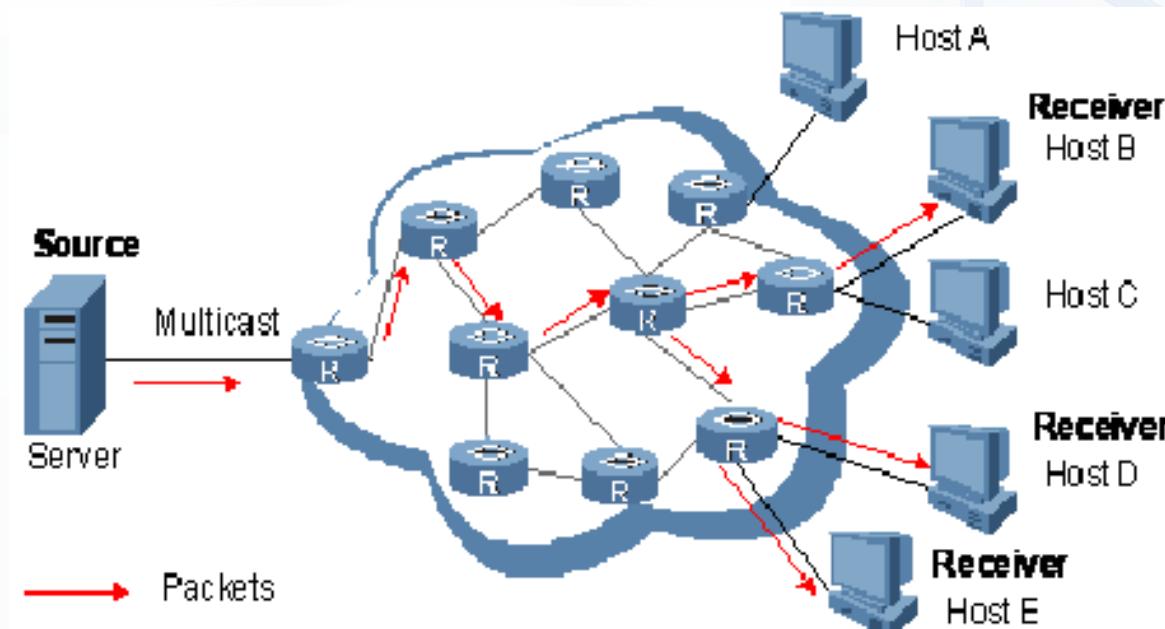
- Opět vychází z IGMPv1 a IGMPv2 a je s nimi zpětně kompatibilní
- Info o opuštění skupiny posílá na 224.0.0.22
- Nově zavádí možnost více zdrojů v jedné skupině
  - Tedy už nemusí být jen jeden zdroj dat, ale může jich být více
  - Nepřipojuji se tedy jen ke skupině, ale i ke konkrétnímu zdroji
  - Nově tedy rozlišujeme dva modely
    - ASM – Any source multicast
      - Sice může být více zdrojů v jedné skupině, ale nerozlišují se
    - SSM – Source specific multicast
      - Může být více zdrojů, které jsou ale nově – díky IGMPv3 rozlišitelné a je tedy možné si zvolit skupiny i zdroj

## IGMPv3 – Joining a Group



# Multicast – Skupinové směrování

- IGMP nám vyřešilo registraci stanic do jednotlivých skupin na lokálních routerech
- Dalším krokem je přenos a směrování data mezi zdrojem a přijímačem
- S běžnými směrovacími metodami zde nevystačíme
  - Protože cílů kam data směruje může být více – dle příslušnosti přijímačů ve skupinách za jednotlivými porty routeru
- Nad sítí, což je zdroj/e, seznam příjemců a seznam routerů, se snažíme vytvořit strom
  - Aby bylo možné společnou cestou posílat data jen jednou
  - Stromy jsou dvojího typu **source-base** a **shared-base**
- Směrovacích protokolů v multicast je více:
  - DVMRP
  - MOSPF
  - PIM-DM
  - PIM-SM
  - CBT

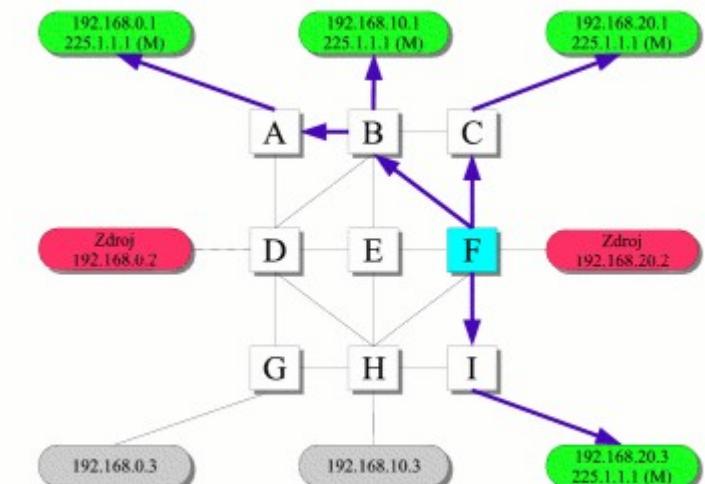
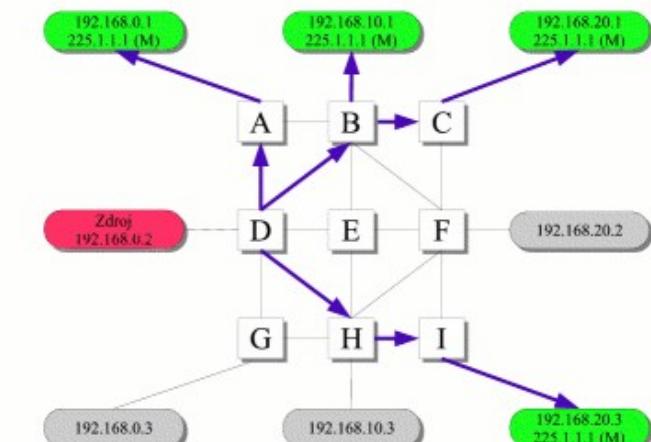


zdroj: [https://www.researchgate.net/figure/Classification-of-Multicast-Routing-Protocols\\_fig2\\_278023165](https://www.researchgate.net/figure/Classification-of-Multicast-Routing-Protocols_fig2_278023165)

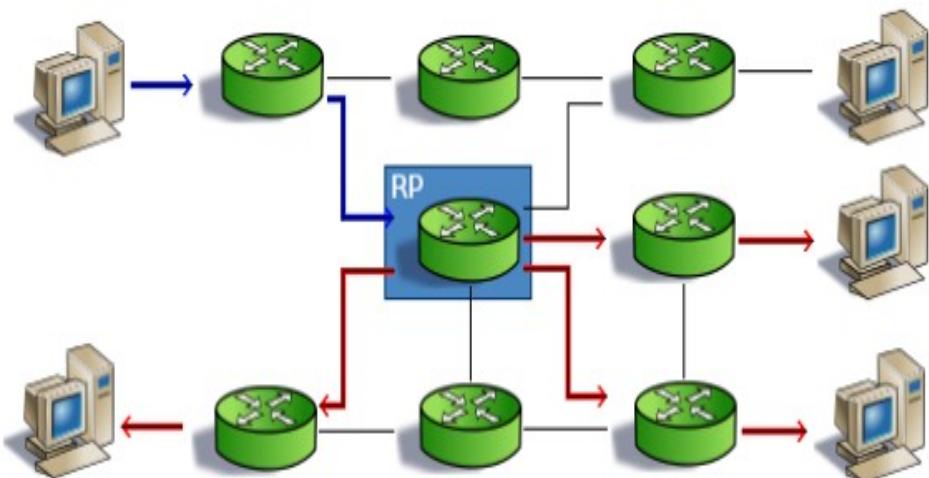
# Multicast – Skupinové směrování: Stromy



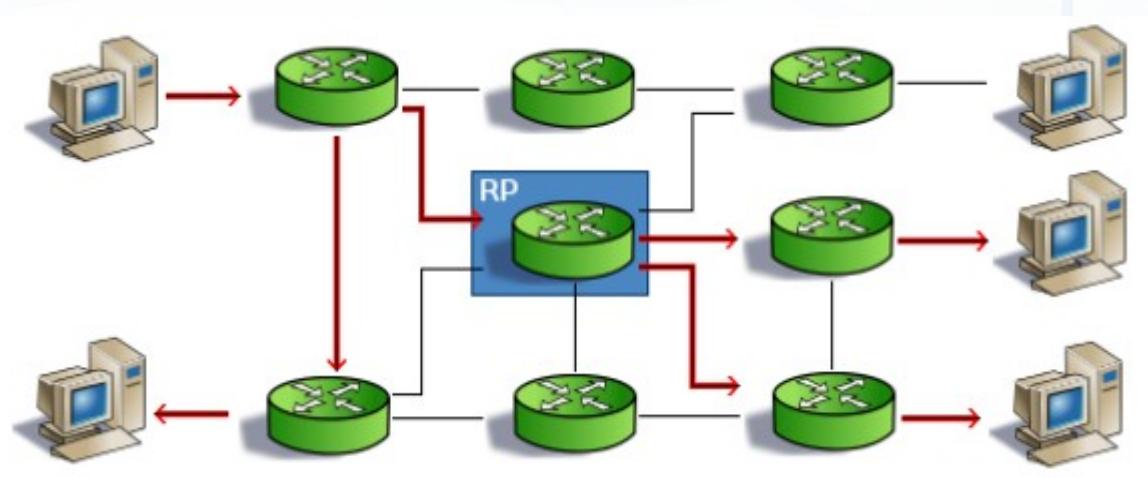
- **Source-base tree** – Source tree – Zdrojový strom
  - Kořen je vysílač, listy jsou příjemci
  - Pro různé zdroje v jedné skupině budou různé stromy
    - Označují se  $(S,G)$ , např  $(192.168.1.1, 225.1.1.1)$
  - Využívají jej Dense mode protokoly ( husté )
- **Shared-base tree** – Shared tree – sdílené stromy
  - Kořen stromu pro danou skupinu je vždy na jednom místě bez ohledu na zdroj dat
  - Kořen je označován jako RP – Rendesvous point
  - Označované jako  $(*, G)$ , např  $(*, 225.1.1.1)$
  - Existuje ve dvou variantách
    - Jednosměrný - data jsou unicasterem poslána na kořen a ten pak zajišťuje jejich distribuci
    - Obousměrný – zdroj posílá data směrem ke kořeni a zároveň směrem k listům
  - Využívají jej Sparse mode protokoly ( řídké )
  - Vzniká problém při výpadku RP



# Multicast – Skupinové směrování: Stromy: Jednosměrný a obousměrný



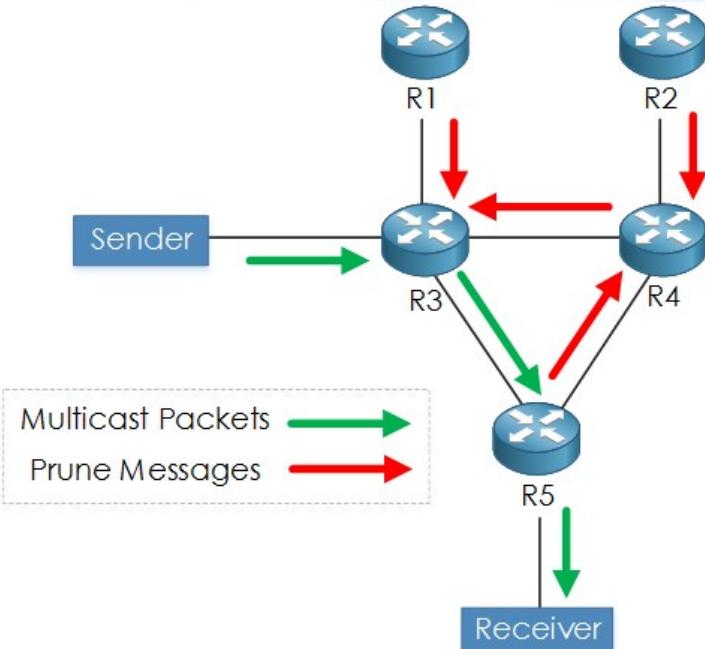
Jednosměrný doručovací strom



Obousměrný doručovací strom

# Multicast – Skupinové směrování: Typy protokolů

- **Dense mode**
  - Využívají zdrojové stromy
    - Kořen stromu je zdroj dat
  - Využívají „push“ principu
    - Primární předpoklad, že data jsou třeba všude – do všech větví stromu
    - Tam, kde jsou větve bez účastníka multicastového provozu, je poslána pro danou větev „prune“ zpráva
      - Ta má jen omezenou platnost
    - Po přijetí „prune“ už nejsou do dané větve posílána data dané multicastové skupiny
      - Větev byla „oříznuta“ - „orezavání větví“
  - Používá se tam, kde předpokládáme, že většina směrovačů bude součástí multicastu - „husté zastoupení“ / „hustý provoz“
    - Nejde o množství dat, ale zastoupení směrovačů
  - Příkladem jsou protokoly DVMRP a PIM-DM
- **Sparce mode**
  - Využívá sdílené stromy
    - Kořen stromu může být libovolný
  - Využívají „pull“ principu
    - Snaží se šetřit přenosy, takže posílá data jen tam, kde jsou požadována
    - Pokud se příjemec přihlásí přes IGMP do multicastové skupiny, router pošle toto info směrem ke kořeni stromu
      - Platnost přihlášení je časově omezena
      - Pokud v dané věti není žádný živý příjemec je „oříznuta“
  - Používá se tam, kde se předpokládá řídké zastoupení směrovačů v multicastové komunikaci – proto „řídký“
  - Příkladem jsou protokoly PIM-SM či CBT

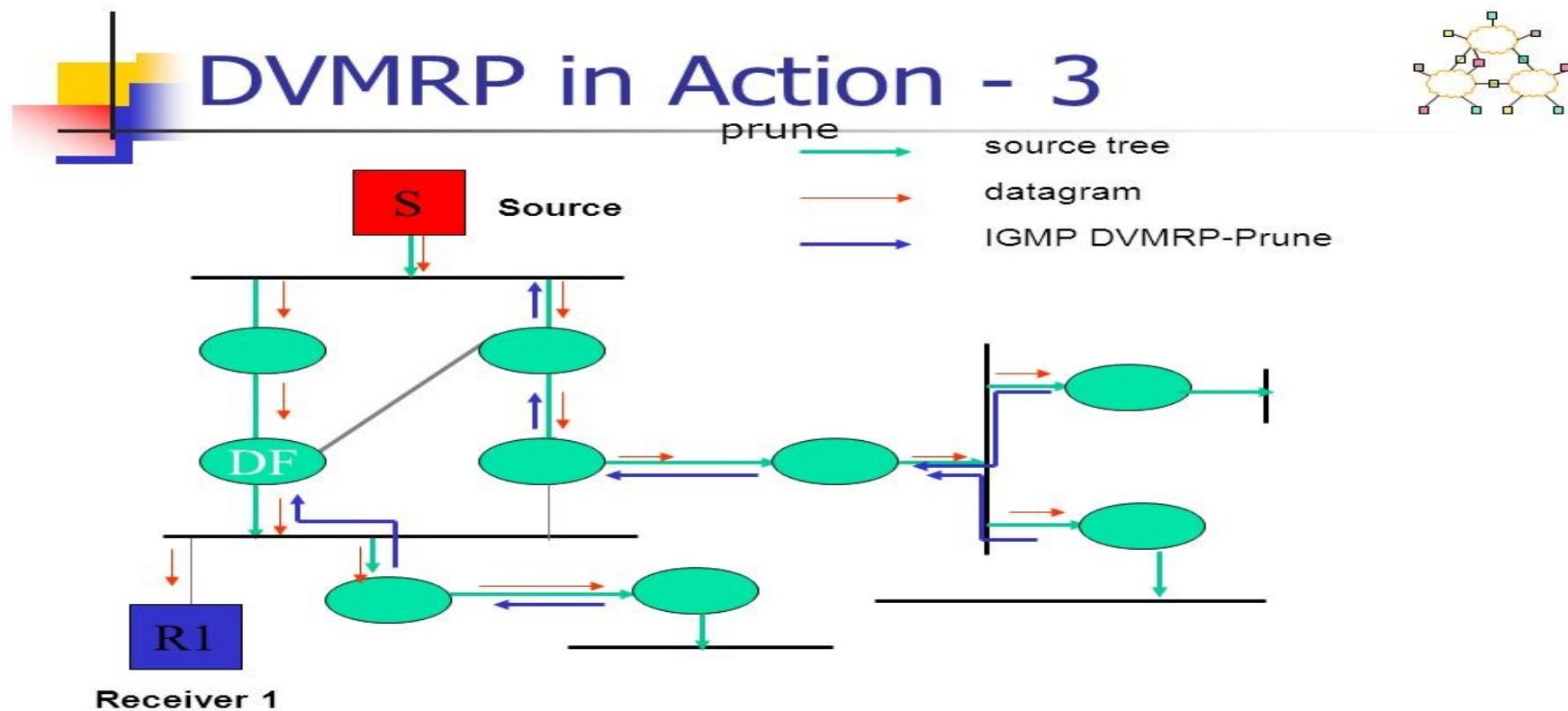


zdroj: <https://www.packetflow.co.uk/what-is-pim-protocol-independent-multicast/>

# Multicast – Skupinové směrování: DVMRP

- DVMRP - Distance Vector Multicast Routing protocol
- Používá source base stromy a tedy se jedná o dense mode protokol
  - „hustý režim“
- Používá záplavové doručování a ořezávání hran
- Vychází z RIP protokolu, ale je výrazně složitější
- Sám si podobně jako RIP sestavuje i unicastovou směrovací tabulku
  - Je v tomhle ohledu nezávislý
- Strom vytváří pomocí Reverse Path Multicasting ( RPM )
  - Pokud přijme zprávu pomocí RPM a v unicastové směrovací tabulce zkontroluje zda se jedná o nejkratší možnou cestu
    - Pokud ano, pošle data dále na všechny rozhraní krom toho odkud přišel
    - Pokud ne, data zahodí
- Pokud v dané podsíti – za routerem - už není žádná živá multicastová stanice, posílá **Prune** a dočasně se odpojuje od stromu

# Multicast – Skupinové směrování: DVMRP: Příklad



# Multicast – Skupinové směrování: MOSPF

- MOSPF – Multicast OSPF
- Používá source base stromy a tedy se jedná o dense mode protokol
  - „hustý režim“
- Vychází a používá OSPF
  - Unicastové
- Každý MOSPF router v paměti drží info o celé topologii sítě
- Do ceny cest – a tedy následně do výpočtu – se zohledňuje i počet stanic na dané cestě
  - Čím více stanic tím lépe – tím více požadavků odbavím jednou kopií dat
- Je to patrně jediný zástupce link state protokolů pro multicast, ale reálně se příliš nepoužívá
  - Problém je v náročnosti přepočtu po každé změně v síti

# Multicast – Skupinové směrování: PIM-DM

- PIM-DM – Protokol Independent Multicast – Dense Mode
- Používá source base stromy a tedy se jedná o dense mode protokol
  - „hustý režim“
    - Stejně jako DVMRP předpokládá, že všichni chtějí přijímat
- Může být použit libovolný směrovací protokol pro zjištění reverzní cesty ( RPF – Reverse Path Forwarding ) pro zjištění nejkratší cesty / eliminaci smyček
- Ve výchozím stavu používá záplavové směrování s následným ořezáváním hran
  - Flood-and-prune
- Existenci ostatních směrovačů zjišťuje pomocí Hello zpráv, které cyklicky každých 30s posílá ostatním směrovačům
  - Tím eliminuje čas, po který by posílal data směrem, kde už nejsou potřeba

# Multicast – Skupinové směrování: PIM-SM

- PIM-SM – Protokol Independent Multicast – Sparse Mode
- Používá shared base stromy a tedy se jedná o sparse mode protokol
  - „řídký režim“
  - Jako RP je použitý router s nejvyšší IP a je označován jako DR -Designated Router
- Používá Join zprávy
- Nalezení reverzní cesty ( RPF ) je nezávislé na konkrétním směrovacím protokolu
- Doručovací stromy se budují mezi příjemcem a RP (Randevous Point) – univerzální (ASM – Any Source Multicast) strom
- Pokud je cesta ke konkrétnímu zdroji kratší, přechází PIM-SM od ASM ke SSM (Source Specific Multicast)

# Multicast – Skupinové směrování: CBT

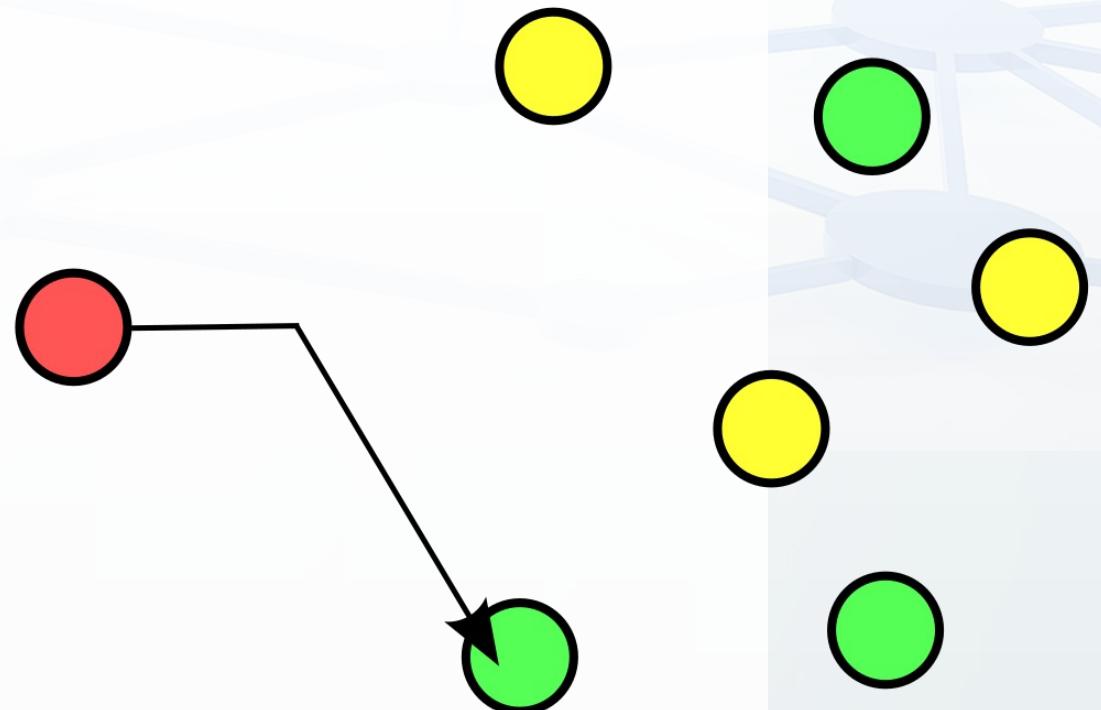
- CBT – Core Base Tree
- Přebírá charakteristiky PIM-SM
- Řídký režim, explicitní připojení, sdílené doručovací stromy
- Efektivnější při vyhledávání zdrojů než PIM-SM
- Rozděluje síť na jednotlivé oblasti ( area ) a vytváří infrastrukturu ( páteř ) pro doručování multicast zpráv
  - V každé oblasti je jeden „páteřní router“
- Není komerčně používán

# Multicast – Směrování :Porovnání směrovacích protokolů

Protocol	Dense Mode?	Sparse Mode?	Implicit Join?	Explicit Join?	(S,G) Source-base tree	(*,G) shared tree?
DVMRP	Yes	No	Yes	No	Yes	No
MOSPF	Yes	No	No	Yes	Yes	No
PIM-DM	Yes	No	Yes	No	Yes	No
PIM-SM	No	Yes	No	Yes	Yes, maybe	Yes, initially
CBT	No	Yes	No	Yes	No	Yes

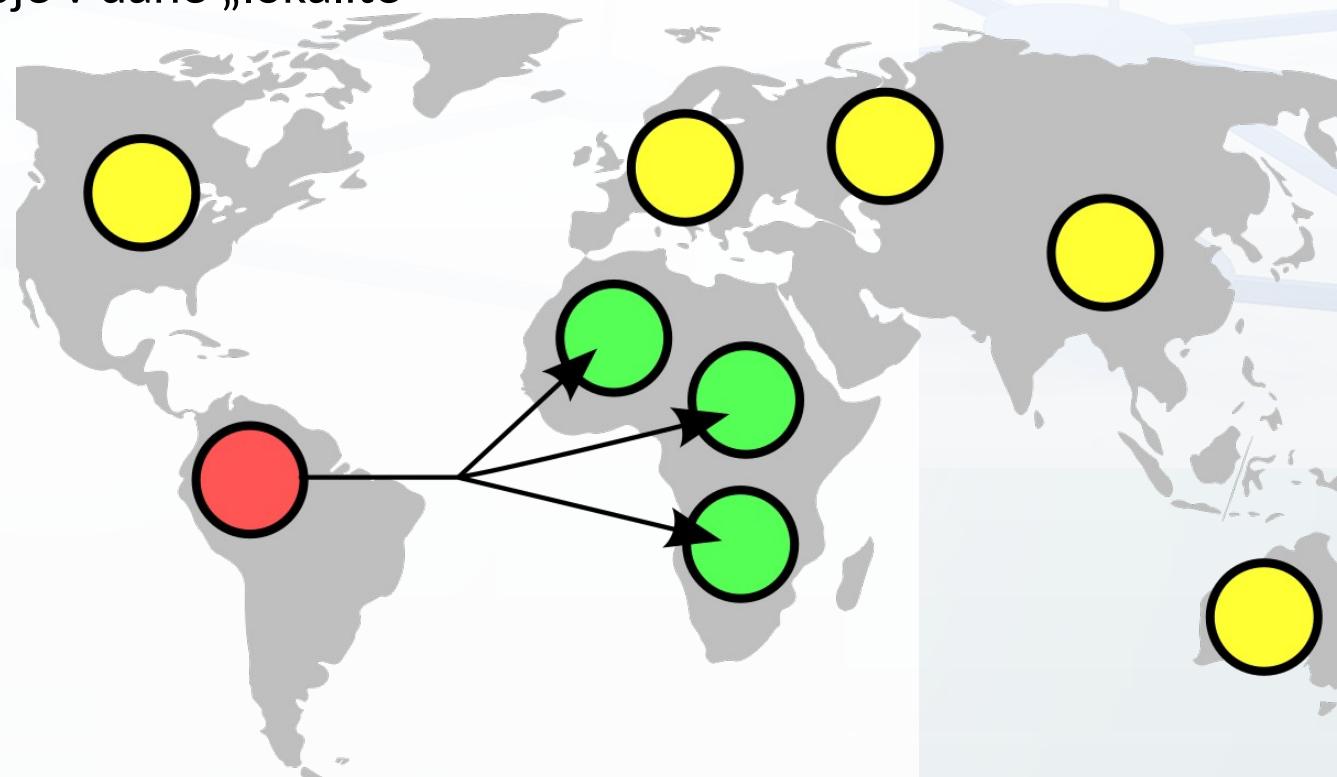
# Anycast

- Specifický typ přenosu dat a směrování, kde cílem není
  - Jeden konkrétní stroj ( unicast )
  - Všechny stroje v dané síti ( broadcast )
  - Všechny stroje v dané skupině ( multicast )
- Cílem je jeden stroj ze skupiny
- Směrování záleží na zadaných konstantách zdrojů a aktuálním stavu zatížení sítě
- Typické použití je připojení na CDN
  - CDN - Content delivery network
    - Sítě poskytující obsah - například obrázky
    - Data jsou typy uložena násobně
    - Je „jedno“, který z dostupných zdrojů použije
    - Vybírám tedy jeden z množiny



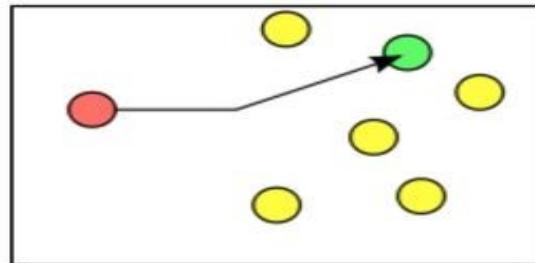
# Geocast

- Specifická varianta multicastu
- Specifický typ přenosu, kde cílem jsou všechny stroje v dané „lokalitě“
  - Tedy například všechna zařízení v Africe
  - Nemusí jít o opravdu všechna, ale může jít o všechna má zařízení v Africe
  - Není nutné vázanou na kontinenty
- Adresa geocastu může mít tři podoby
  - Bod
  - Kruh se středem a poloměrem
  - Mnohoúhelník – defakto seznam bodů

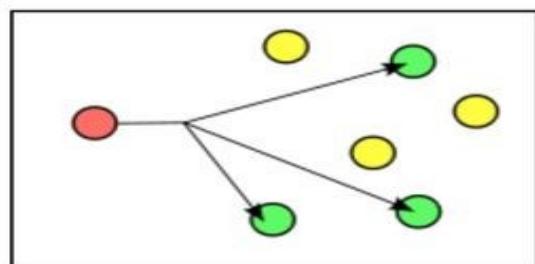


# Porovnání typů přenosů

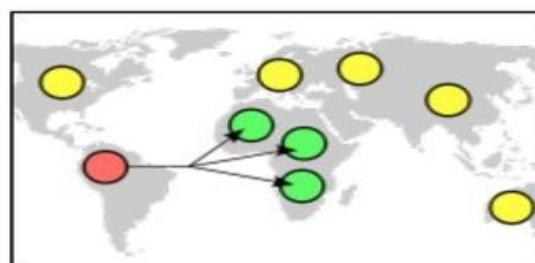
## Unicast, Broadcast, Multicast, Anycast



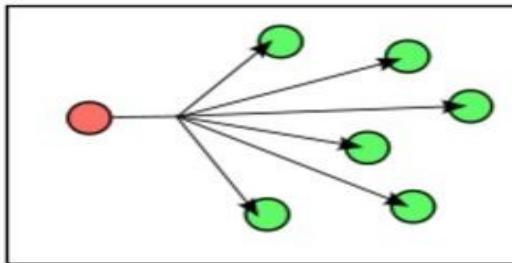
Unicast:  
One specific  
receiver



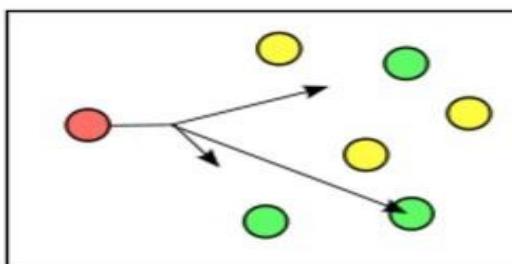
Multicast:  
Many receivers,  
all of a specific  
group



Geocast:  
Many receivers,  
all of a geographic region



Broadcast:  
Many receivers,  
all on the network

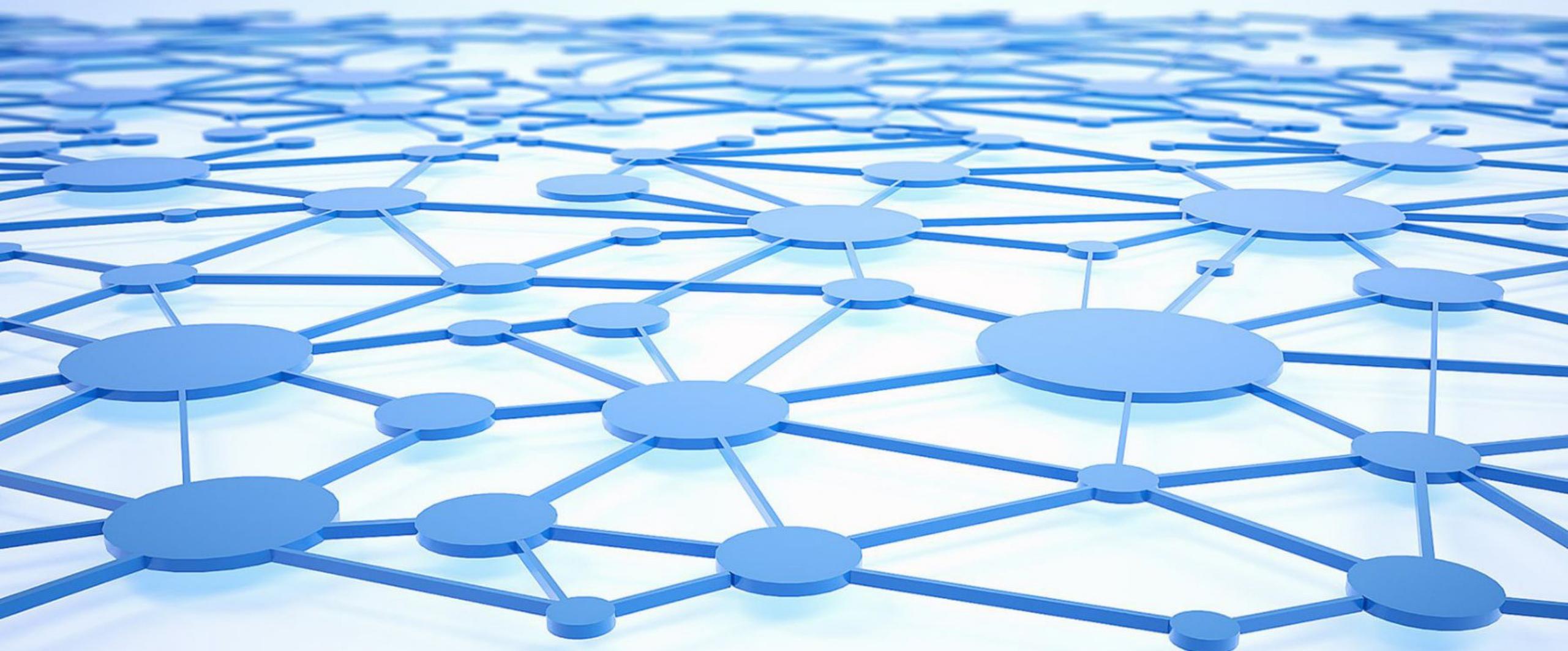


Anycast:  
One receiver,  
"nearest" of a  
specific group

Pictures: Wikipedia

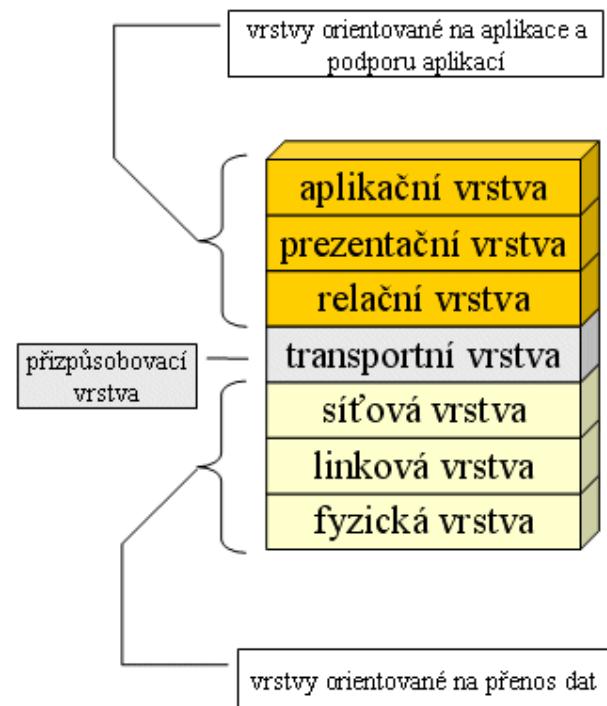
# Úvod do počítačových sítí

Přednáška 10 ( 2025/2026 )  
ver. 2025-11-11-01



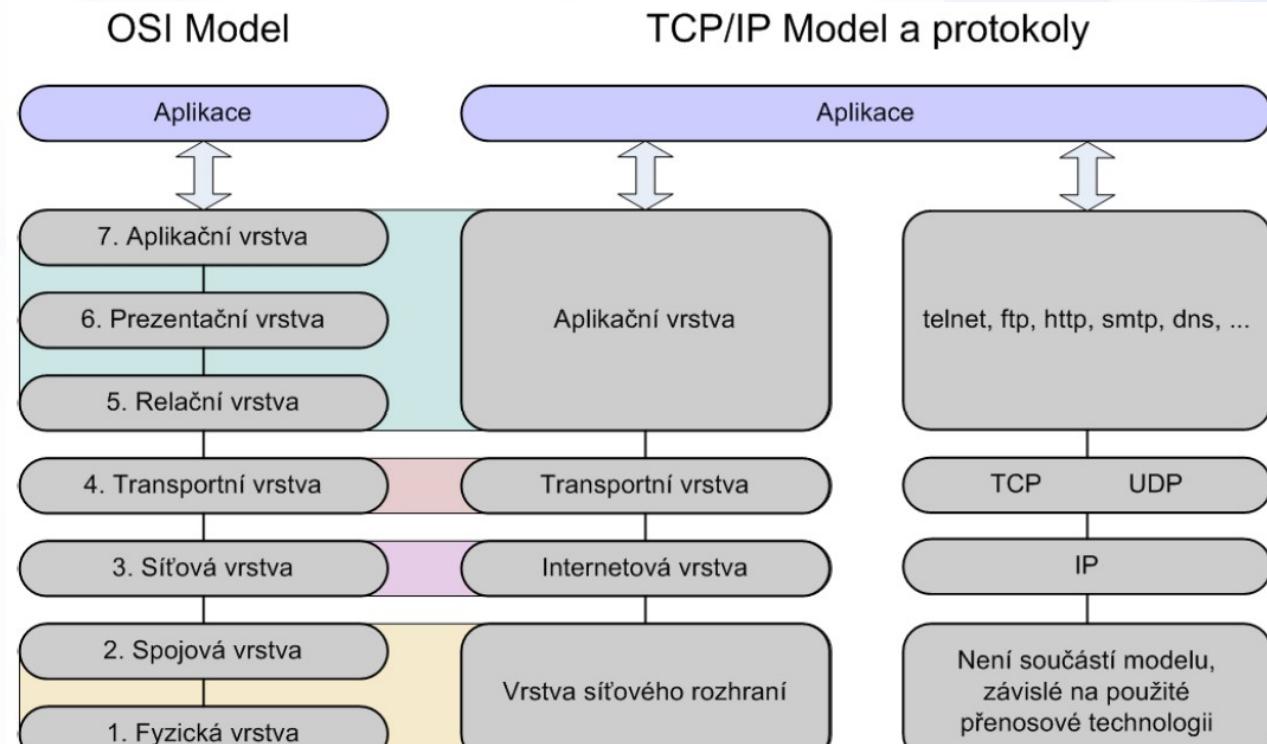
# L4 – Transportní vrstva

- Základní funkce L4
  - Přijímá data od aplikačně orientovaných vrstev a předává je nižším vrstvám
    - Poskytuje vyšším L5-L7 vrstvám služby, které umožňují přenos pomocí L1-L3 bez znalosti jejich fungování
  - Zajišťuje „přizpůsobení“
    - „proxy“ mezi požadavky vyšších vrstev a možnostmi nižších vrstev
  - Zajišťuje multiplexing a demultiplexing
  - Zajišťuje end-to-end komunikaci
    - Komunikaci mezi aplikacemi
- Volitelné / rozšiřující funkce L4
  - Řízení toku dat
  - Podpora QoS
  - Předcházení zahlcení



# Porovnání transportní vrstvy pro ISO/OSI a TCP/IP model

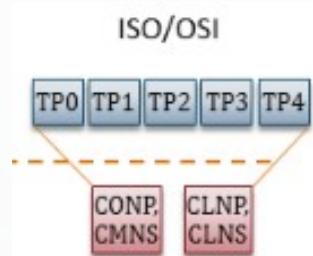
- V obou modelech je transportní vrstva zastoupena a na stejném úrovni - L4
- Provedení L4 se mezi ISO/OSI a TCP/IP výrazně liší
  - Provedením se myslí původní plán / návrh
- Dnes už se reálně používá jen TCP/IP varianta
- ISO/OSI nabízí vyšším vrstvám 5 variant přenosů
  - T0, T1,T2,T3,T4 – transportní třídy
  - Rovnou počítá s více možnostmi
  - Dnes už se nepoužívá
- TCP/IP nabízí v základu 2 varianty
  - UDP a TCP
  - Další se mohou postupně přidávat



Obr. 10: Srovnání modelu RM ISO/OSI a TCP/IP

# L4 protokoly v ISO/OSI

- ISO/OSI počítalo s 5 druhy služeb poskytovaných vyšším vrstvám
- Vychází z původního návrhu L3 ISO/OSI, které může fungovat :
  - Spolehlivě a spojovaně
    - CONP – Connection Oriented Network Protokol
    - Přenosový protokol
  - CMNS – Connection Mode Network Service
    - Služba pro vyšší vrstvy
- Nespolehlivě a nespojovaně
  - CLNP – Connectionless Network Protokol
  - CLNS – Connectionless Network Service
- TCP/IP L3 umí jen jednu variantu – IP



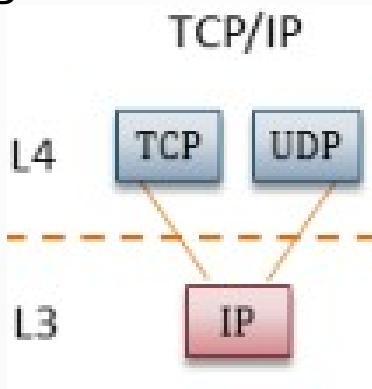
zdroj: <https://www.eearchiv.cz/l226/slides.php?l=9&me=3>

Služba	TP0	TP1	TP2	TP3	TP4
Spojované služby síťové vrstvy	✓	✓	✓	✓	✓
Nespojované služby síťové vrstvy	✗	✗	✗	✗	✓
Sřetězování a rozdělování	✗	✓	✓	✓	✓
Segmentace a skládání segmentů	✓	✓	✓	✓	✓
Oprava chyb	✗	✓	✗	✓	✓
Znovunavázání spojení (při velkém množství nepotvrzených PDU)	✗	✓	✗	✓	✗
Multiplexování a demultiplexování jediným virtuálním okruhem	✗	✗	✓	✓	✓
Explicitní řízení toku dat	✗	✗	✓	✓	✓
Opakování vysílání při prodlevě	✗	✗	✗	✗	✓
Spolehlivá transportní služba	✗	✓	✗	✓	✓

zdroj: [https://cs.wikipedia.org/wiki/Transportn%C3%AD\\_vrstva](https://cs.wikipedia.org/wiki/Transportn%C3%AD_vrstva)

# L4 protokoly v TCP/IP

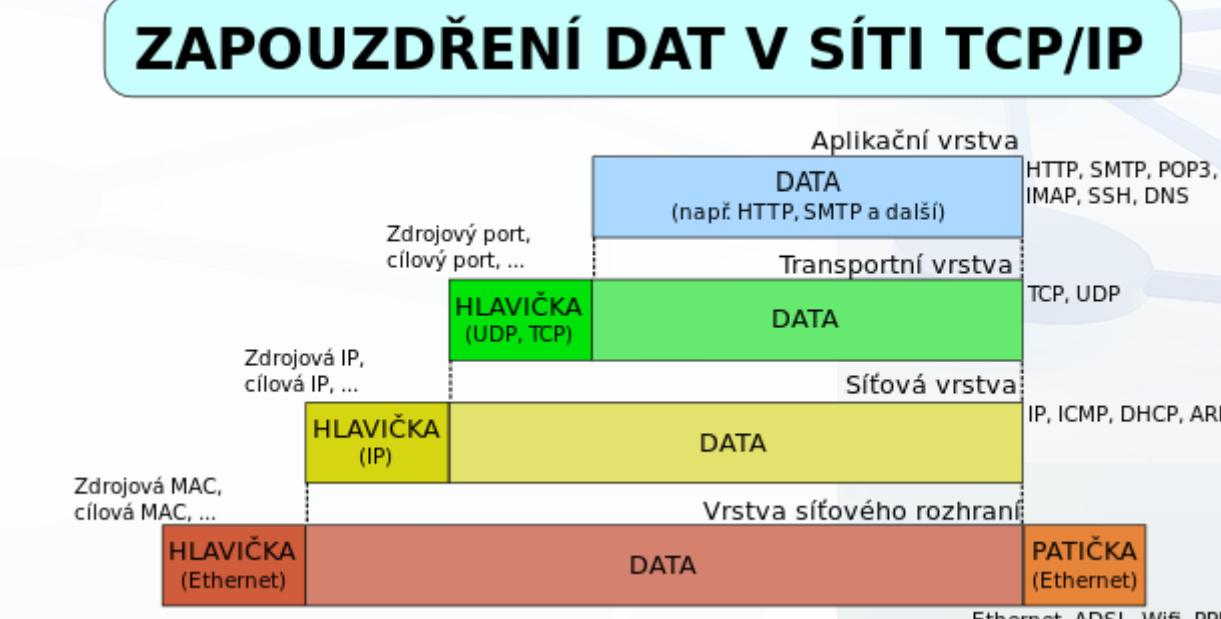
- TCP/IP může na L3 využít jen IP
  - Nespojovaný a nespolehlivý
- V základu poskytuje dva základní protokoly
  - Spolehlivě a spojovaně - TCP
  - Nespolehlivě a nespojovaně – UDP
- Postupně doplněné o další řešící nedostatky dvou základních
  - SCTP, DCCP, RUDP



<i>Protokol UDP</i>	<i>Protokol TCP</i>
Mezi počítači není třeba vytvořit relaci.	Mezi počítači se vytváří relace.
Nezaručuje doručení dat, nezajišťuje potvrzování ani správné pořadí dat.	Zaručuje dodání paketů s využitím potvrzování a správného pořadí při přenosu dat.
Programy používající UDP musí zajistit spolehlivost přenosu dat.	Programům používajícím TCP je zaručen spolehlivý přenos dat.
Je rychlý, s malými požadavky na objem provozních dat, podporuje dvoustrannou komunikaci a komunikaci jednoho počítače s více počítači.	Je pomalejší, má vyšší nároky na objem provozních dat, podporuje pouze dvoustrannou komunikaci.

# Základní funkce transportní vrstvy: Zapouzdření dat

- Základní přenosovou jednotkou L4 je segment
- Segment dostane L4 od vyšších vrstev L5-L7
- Segment vkládáme jako data do L3 paketu
- Segment má hlavičku a data
  - Hlavička se liší dle konkrétního protokolu
    - Tedy je jiná pro TCP i UDP
    - Obsahuje L4 adresaci + další meta data
      - Například checksumu – zabezpečení se tedy může řešit i na L4
        - Zde se např u UDP a TCP počítá s „pseudo-hlavičkou“
        - Cílem je odhalení podvržených dat
        - Checksumu zde musí přepočítávat i NAT
  - Data jsou přijatá data od vyšších vrstev



zdroj:  
[https://cs.wikipedia.org/wiki/Zapouzd%C5%99en%C3%A1\\_\(po%C4%8D%C3%ADta%C4%8Dov%C3%A9\\_s%C3%ADt%C4%9B%C4%9B\)](https://cs.wikipedia.org/wiki/Zapouzd%C5%99en%C3%A1_(po%C4%8D%C3%ADta%C4%8Dov%C3%A9_s%C3%ADt%C4%9B%C4%9B)

# Základní funkce transportní vrstvy: Zapouzdření dat- TCP a UDP hlavičky

**TCP Segment Header Format**

Bit #	0	7	8	15	16	23	24	31				
0	Source Port				Destination Port							
32	Sequence Number											
64	Acknowledgment Number											
96	Data Offset	Res	Flags		Window Size							
128	Header and Data Checksum				Urgent Pointer							
160...	Options											

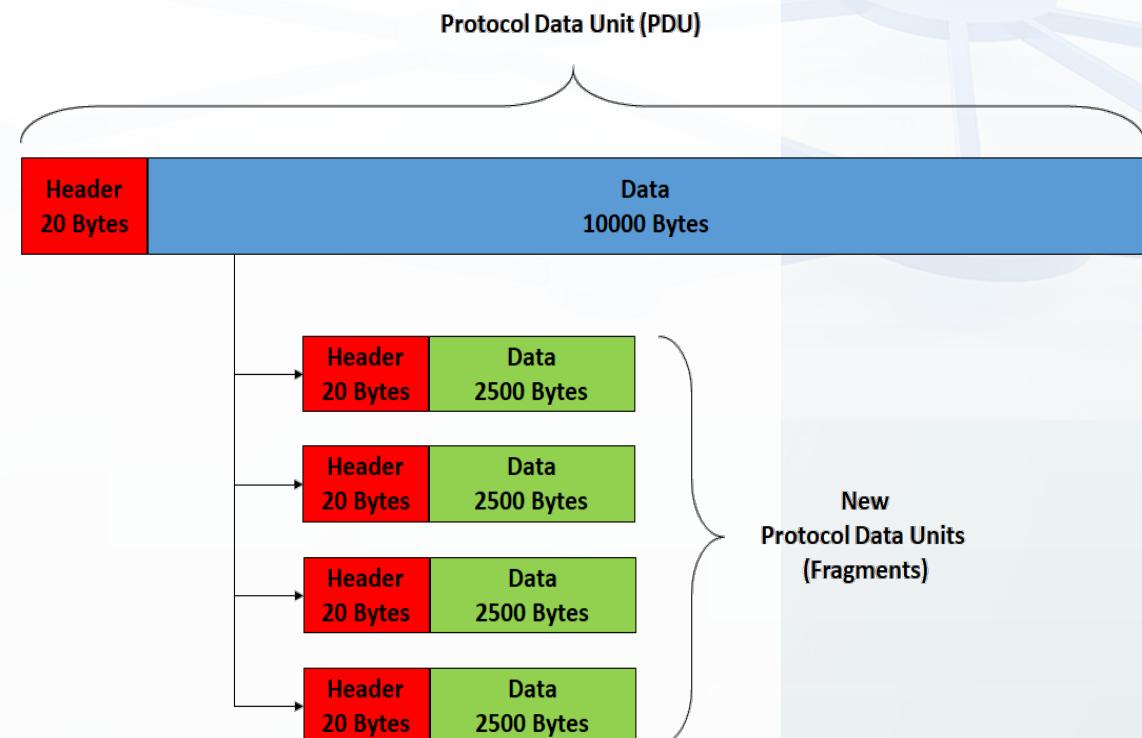
**UDP Datagram Header Format**

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

# Segmentace a fragementace dat:

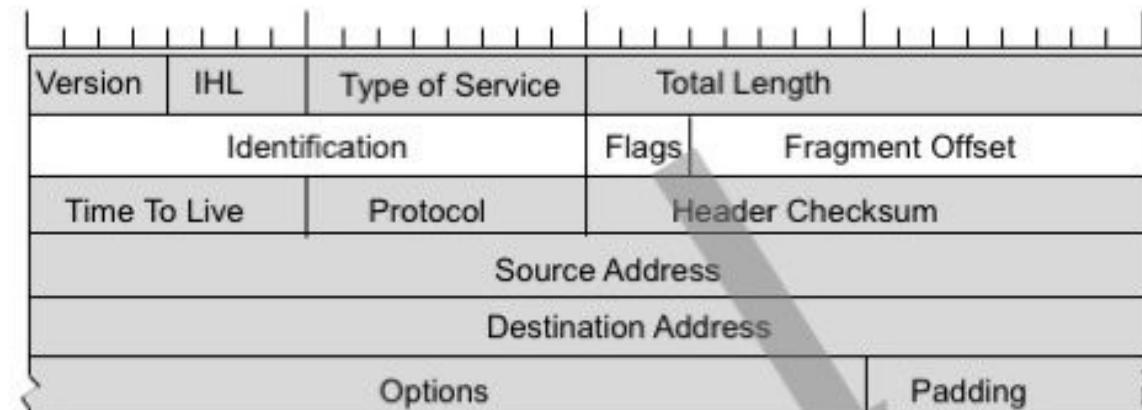
## Základní princip

- Každá vrstva má svoji přenosovou jednotku a ta má omezení z hlediska velikosti
  - L2 – rámec
    - MTU Ethernetu 1500 nebo 1492 bytů
      - Nebo 9000 bytů v případě Jumbo Frame
  - L3 – paket
    - IP MTU 65.535 bytů
  - L4 – segment
    - MSS – Maximum Segment Size – definovaná v rámci OS
- Fragmentace umožňuje rozdělit základní jednotku na menší a přenést data postupně
- Fragmentace vyžaduje další režii
  - Dělení a opětovné složení dat není zadarmo
- U příjemce proběhne defragmentace – sestavení původních dat
  - Data mohou přijít mimo pořadí
    - Řeší TCP tím, že si v bufferu data umí přeskládat dle potřeby



# Segmentace a fragementace dat: Povolení a zákaz fragmentace

- K fragmentaci dojde, pokud potřebuji větší data vyšší vrstvy přenést v menší jednotce nižší vrstvy
  - MTU je 1500, potřebuji přenést 6000
    - Segment bude přenesen na 4 části po 1500 bytech
- Pokud má k fragmentaci dojít, musí být podporována – především na L3 a L4
- Zda je fragmentace povolena či nikoliv, je určeno v IP záhlaví v rámci DF bitu
  - Pokud je fragmentace povolena, budou větší data rozdělena
  - Pokud není povolena, bude odesílatel informován, že data nelze přenést
- Druhý důležitý flag je MF bit, který nabývá dvou hodnot
  - 1 pokud data jsou fragmentována a nejedná se o poslední fragment
  - 0 pokud se jedná o poslední fragment dat

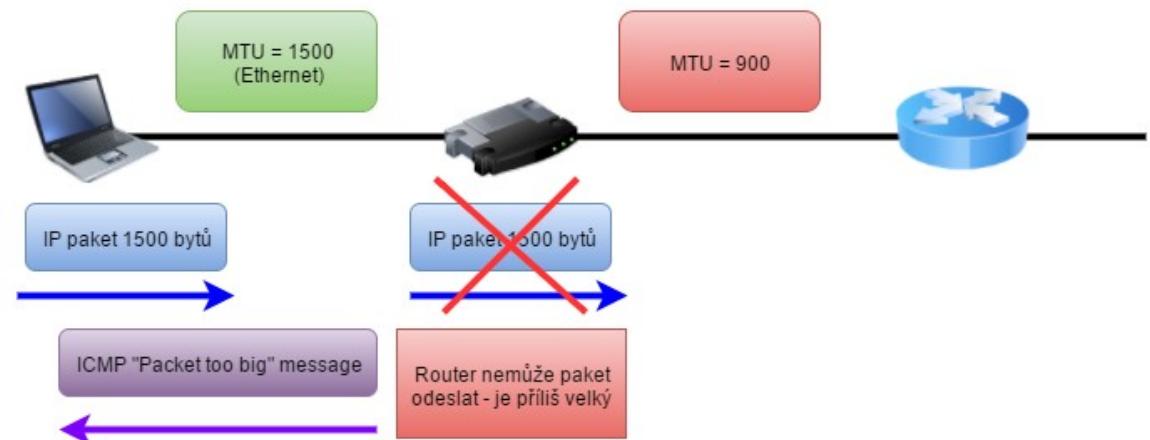


Flags: bit 0 – Reserved  
bit 1 - Don't Fragment  
bit 2 – More Fragments

# Segmentace a fragementace dat: PMTUD

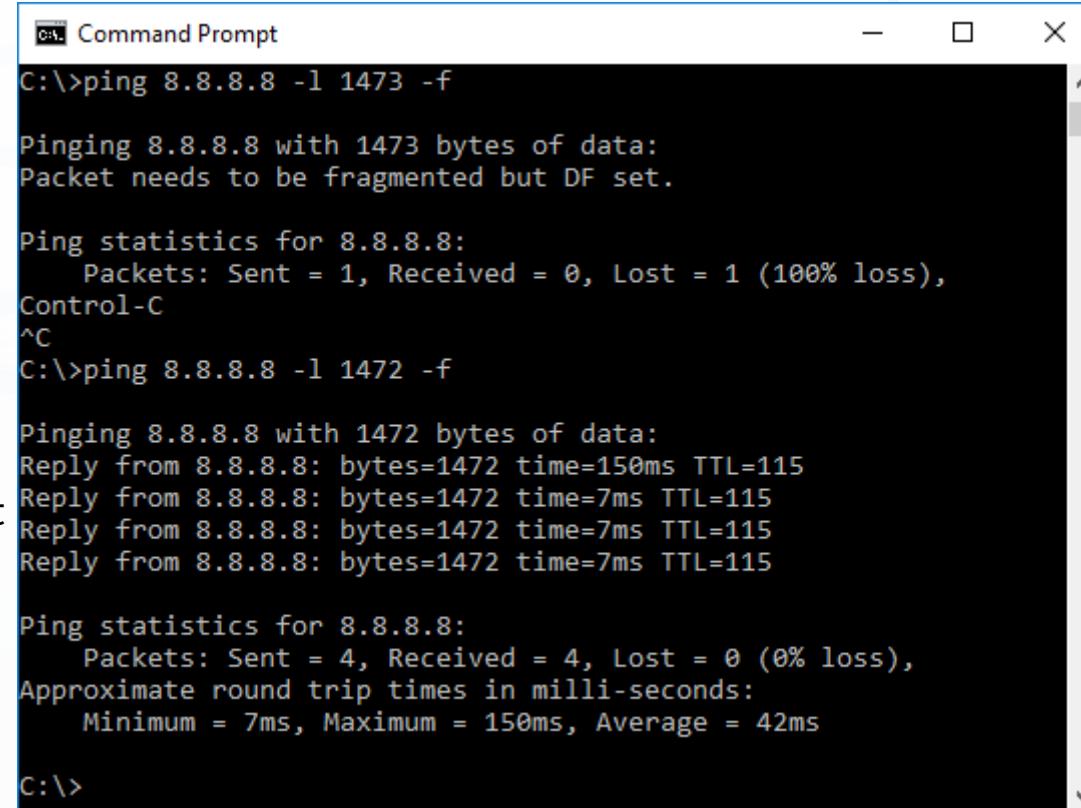


- Na příkladu z obrázku se snažíme poslat data větší než mohou routerem projít
- Zde funguje PMTUD – Path MTU Discovery
  - Snažím se najít optimální velikost dat
  - Na prvním skoku nastavení MTU znám, ale dále nevím
    - Znám, protože zde je přímé spojení
    - POZOR i zde mohou mít obě proti strany různé nastavení
    - POZOR nemusí se to nutně poznat na běžném pingu
      - Protože posílá příliš malá data
  - Pošlu tedy data v maximální velikosti podle lokální
    - A nastaveným DF bitem na 1 – NE fragmentovat
  - Pokud data projdou – OK
    - Zvyšovat velikost nemůžu – má MTU to nedovolí
  - Pokud data někde cestou neprojdou, vrátí se ICMP zpráva „Packet too big“
  - Odesílatel pak sníží velikost a zkusí to znova
    - A to opakuje dokud data neprojdou
  - Musí být povolené ICMP



# Segmentace a fragmentace dat: Problémy s fragmentací

- Kde to jde, snažíme se fragmentaci vyhnout
  - Například tak, že nenačítáme/neposíláme v aplikace větší data než je MTU
- Nevýhody fragmentace
  - Zvýšená režie
    - To rozdelení a opětovné složení dat stojí čas a paměť
  - Zvýšená pravděpodobnost chyby
    - Pokud je jeden segment/fragment špatně, musím přenos opakovat
  - Zavádí stavovost do jinak bezstavového IP
    - Což znamená další režii
      - Paměťovou, protože segmenty musím uchovávat před zpracováním
      - Časovou/přenosovou – musím řešit potvrzování + timeouty
        - » Jako jsme si to říkali na L2



```
C:\>ping 8.8.8.8 -l 1473 -f
Pinging 8.8.8.8 with 1473 bytes of data:
Packet needs to be fragmented but DF set.

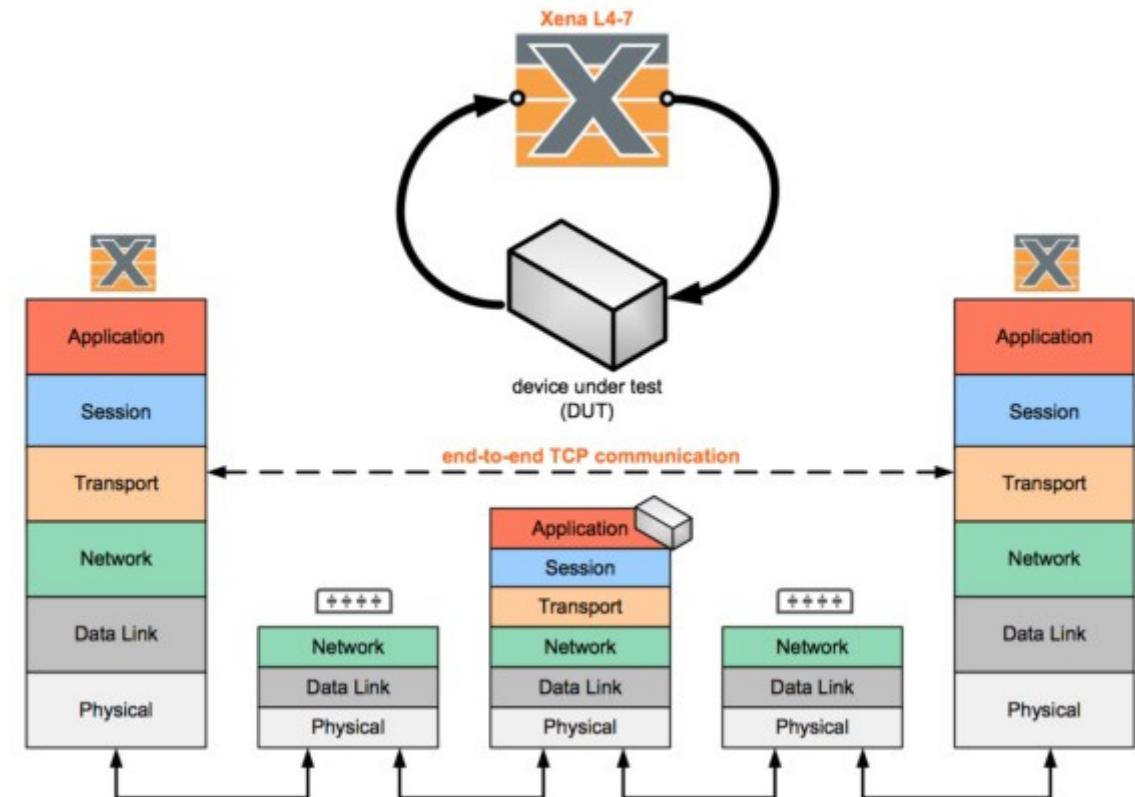
Ping statistics for 8.8.8.8:
  Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
Control-C
^C
C:\>ping 8.8.8.8 -l 1472 -f
Pinging 8.8.8.8 with 1472 bytes of data:
Reply from 8.8.8.8: bytes=1472 time=150ms TTL=115
Reply from 8.8.8.8: bytes=1472 time=7ms TTL=115
Reply from 8.8.8.8: bytes=1472 time=7ms TTL=115
Reply from 8.8.8.8: bytes=1472 time=7ms TTL=115

Ping statistics for 8.8.8.8:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
  Minimum = 7ms, Maximum = 150ms, Average = 42ms
C:\>
```

zdroj: <https://networkdirection.net/articles/network-theory/mtu-and-mss/use-ping-to-find-the-mtu/>

# End-to-end komunikce

- L2 dovoluje adresovat v rámci LAN pomocí MAC
  - Identifikuje konkrétní zařízení / sovou kartu / port
- L3 dovoluje adresovat pomocí IP v rámci více sítí
  - Identifikuje konkrétní rozhraní v konkrétním PC
  - Na jednom rozhraní může být i více adres
    - Odděluje je od sebe
- L4 dovoluje adresovat konkrétní proces / aplikaci
  - V rámci jednoho zařízení může fungovat více aplikací
    - Až k zařízení nás dovede L2+L3
    - Ke konkrétní aplikaci nás dovede L4 adresace
      - Protokol + port
- Při přenosu nemusí všechna zařízení rozumět všem vrstvám
  - L4 „stačí na koncových bodech“
  - L2 switche nevidí ani segment ani paket
  - L3 nemusí vidět segment
    - Ale mohou a využívá se toho pro QoS nebo zabezpečení
      - Preferenci či filtrace na základě identifikace aplikace pomocí portu a protokolu

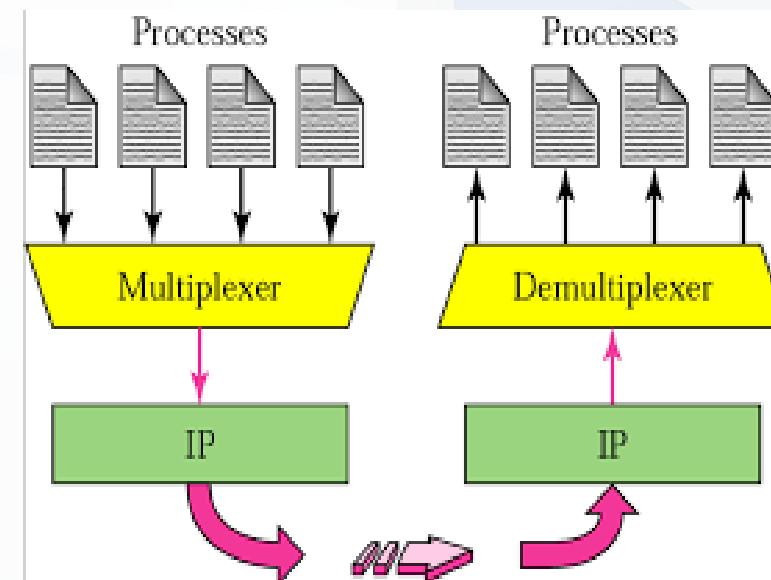


zdroj:

<https://xenanetworks.com/?knowledge-base=knowledge-base/vulcan/vulcanmanager/overview>

# Multiplex a demultiplex na L4

- Multiplex na úrovni L4
  - Data z různých aplikací, ale jediného stroje, jsou přenášena stejnou L3 cestou
  - Na straně odesílatele jsou L4 data – segment – zapouzdřena do paketu a přenesena k příjemci
- Demultiplex na úrovni L4
  - Na straně příjemce je z L3 paketu „vybalen“ segment
  - Segment se může skládat s více částí, takže je vybalen z více paketů a složen dohromady
  - Na základě portu a protokolu je segment-L4 data-předán konkrétní aplikaci
  - Na každé kombinaci IP(L3) + protokol(L4) + port(L4) může „naslouchat“ jen jediná aplikace
  - Ověříme přes netstat -apnl / ss -apnl



# Multiplex a demultiplex na L4: Naslouchání aplikací

```
alf@server:~$ netstat -atu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:ssh              0.0.0.0:*
tcp      0      0 server:ssh               192.168.1.48:59476 ESTABLISHED
tcp6     0      0 [::]:mysql              [::]:*
tcp6     0      0 [::]:ssh                [::]:*
udp      0      0 0.0.0.0:slingshot        0.0.0.0:*
udp      0      0 0.0.0.0:7091             0.0.0.0:*
udp      0      0 0.0.0.0:bootpc           0.0.0.0:*
udp6     0      0 [::]:25087              [::]:*
udp6     0      0 server:dhcpv6-client    [::]:*
udp6     0      0 [::]:59809              [::]:*
[alf@server ~]$ 
```

# Adresace v transportní vrstvě

- Na L2 a L3 jsou adresovali jednoznačně adresou
  - L2 MAC - f8:b1:56:d7:77:f7
  - L3 IP - 10.0.0.100(IPv4) | fe80::208:60ff:fe00:63c8(IPv6)
- Na L4 je „adresa“ složena ze dvou částí:
  - Protokol
    - L4 může podporovat více protokolů – TCP či UDP - např
  - Dle rozlišení TCP/IP a ISO/OSI
    - TCP/IP - Port
      - Kladné celé číslo v rozmezí 0 až 65535
    - ISO/OSI – SAP - Service Access Point
      - Nepoužívá se – jen pro úplnost
- Takže adresace v rámci daného stroje, pak vypadá např TCP/80
- Jeden proces může poslouchat i na více portech i IP
  - Na obrázku nginx
- Na stejném portu, ale jiném protokolu nebo IP mohou poslouchat různé procesy

tcp	0	0	192.168.1.18:8010	0.0.0.0:*	NASLOUCHÁ	13871/nginx: worker
tcp	0	0	127.0.0.1:8010	0.0.0.0:*	NASLOUCHÁ	13871/nginx: worker
tcp	0	0	192.168.1.18:80	0.0.0.0:*	NASLOUCHÁ	1745/apache2
tcp	0	0	192.168.1.18:8880	0.0.0.0:*	NASLOUCHÁ	13871/nginx: worker
tcp	0	0	192.168.1.18:8081	0.0.0.0:*	NASLOUCHÁ	1311/lighttpd

zdroj: Překvapivě vlastní obrázek ;)

# Adresace v transportní vrstvě: Kategorizace portů



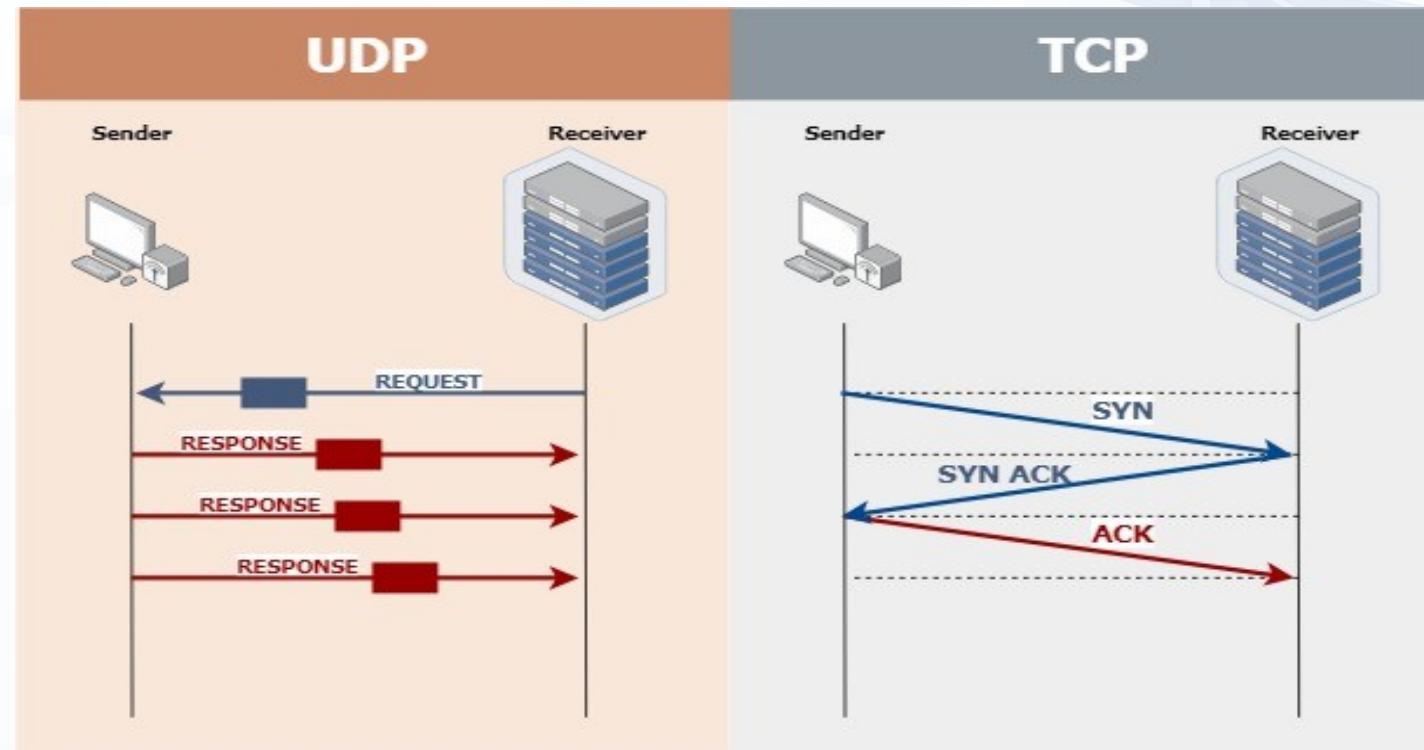
- Obecně může jakýkoliv proces fungovat na jakémkoliv portu – s výjimkou portu 0
- „0“ se chová jinak než ostatní port – v rámci přenosu se nula nahrazuje za jakýkoliv volný v z dynamických portů
- Rozlišujeme tři skupiny portů
  - Dobře známé porty 0 – 1023
    - Na těchto portech běží typicky vždy stejné služby
    - Porty 1 – 1023 jsou označované jako „privilegované“
      - Může se na ně „nabindovat“ pouze administrátor
  - Registrované porty 1024 – 49151
    - Tyto porty mohou být vázány s nějakou typickou službou
      - Ale nemusí
    - Použít je může libovolný uživatel
  - Dynamické porty 49152 – 65535
    - Tyto porty nejsou nejsou nijak specificky významné
    - Typicky se používají pro odchozí spojení
      - Zadám odchozí port nula a on se transformuje na volný port z tohoto rozsahu

el		UDP	TCP
Port #	Popis	Port #	Popis
21	FTP	21	FTP
23	Telnet	23	Telnet
25	SMTP	25	SMTP
69	TFTP	69	TFTP
70	Gopher	70	Gopher
80	HTTP	80	HTTP
88	Kerberos	88	Kerberos
110	POP3	110	POP3
119	NNTP	119	NNTP
143	IMAP	143	IMAP
161	SNMP	161	SNMP
443	HTTPS	443	HTTPS
993	IMAPS	993	IMAPS
995	POP3S	995	POP3S

je-li to možné, je konvence  
stejná pro UDP i TCP !!!

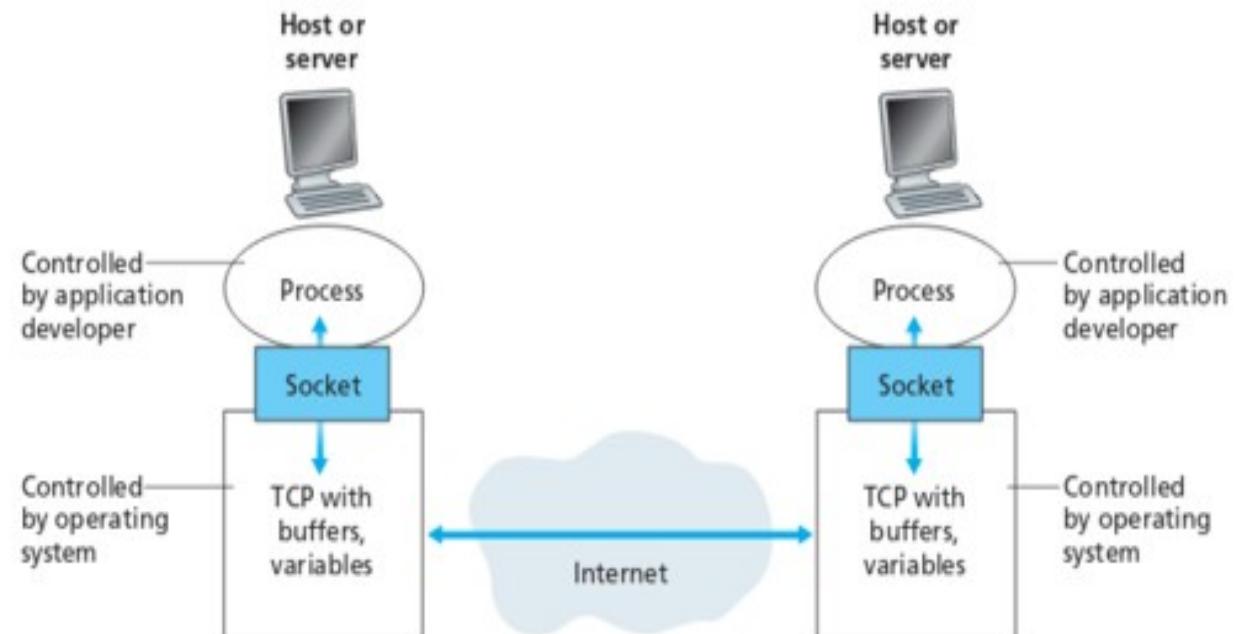
# Protokoly v L4 pro TCP/IP

- Jak už bylo zmíněno ISO/OSI přenosové protokoly se už nepoužívají a používá se výhradně TCP/IP transportních protokolů
  - Ale běžně se uvádí „L4 ISO/OSI protokol je například TCP či UDP“
- Dva základní přenosové protokoly TCP/IP jsou
  - UDP
    - User Datagram Protokol
  - TCP
    - Transmission Control Protocol
- L4 přenosy pomocí TCP či UDP protokolu jsou typicky realizovány nad sockety



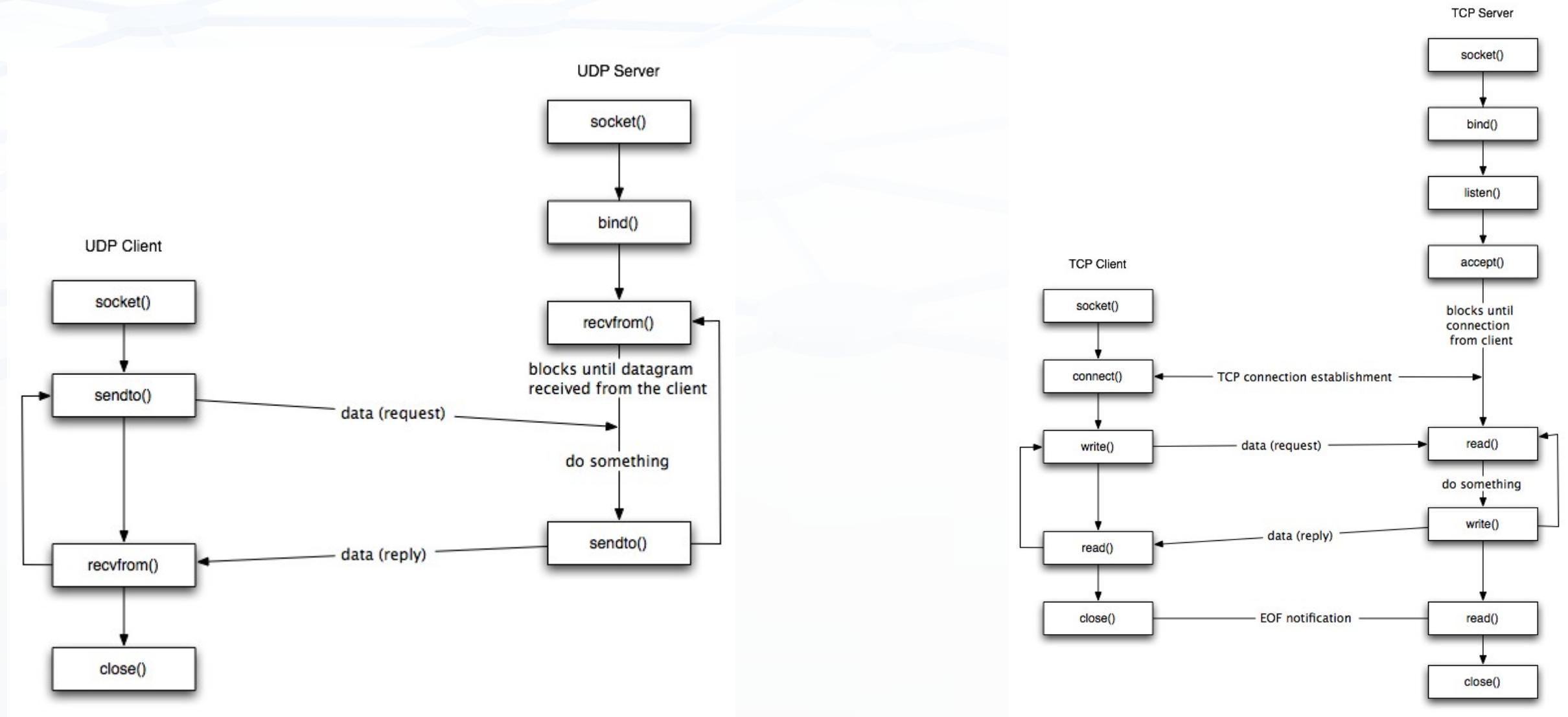
# Berkeley Sockety

- Jedná se o programátorský prostředek umožňující realizovat L4 přenosy např pomocí TCP / UDP
- Původně vychází ze souborů – defakto se jedná o handler, přes který se dá komunikovat
- Implementace je platformě závislá, ale poskytuje platformně nezávislý přístup ke komunikaci
  - BSD sokety, WinSokety, sokety v Javě
    - Programátorské volání se liší, ale jsou vzájemně kompatibilní
- Existuje více typů
  - AF\_Unix – pojmenované sokety
    - Adresou je cesta k souboru
    - Použití jen v rámci daného stroje
    - Nižší režie než TCP/IP a „bezpečnější“
  - AF\_INET – internetové sokety
    - Adresou je IP + port + protokol
    - Možnost realizovat TCP i UDP



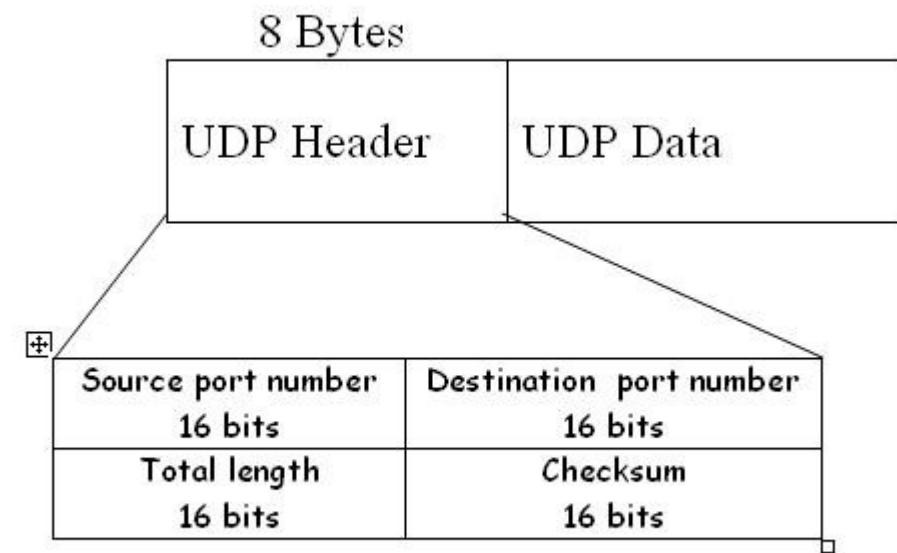
zdroj: <https://stackoverflow.com/questions/152457/what-is-the-difference-between-a-port-and-a-socket/152863>

# Sockety: Porovnání UDP a TCP volání



# Prokoly v L4: UDP

- UDP – User Data Protocol
- Nespojovaný
  - Nenavazuje se spojení – vytváří se samostatný datagram, který je odeslán do sítě
- Nepotvrzovaný
  - Úspěšné doručení jednotlivých datagramů se nepotvrzuje
    - Když se data přenesou správně – „Fajn“
    - Když dojde k chybě, neřeším to, data zahodím a pokračuji dále
    - Stejně tak pokud data přijdou mimo požadované pořadí jsou chybná zahozena a pokračuje se v přenosu dále
- Nenáročný na režii
- Typicky se používá tam, kde „nevadí“ ztráta části dat
  - Např multimedialní přenosy – opakované posílání jednoho ztraceného framu by nadělalo viditelnější škody než jeho vypuštění
- I nad UDP lze realizovat spolehlivý přenos, ale musí jej řešit vyšší vrstva
  - Např. OpenAFS
  - Problém je samozřejmě „cena“ přenosu – tedy režie
    - Logicky, protože o problému se dozvíme až velice vysoko v rámci ISO/OSI a tedy i pozdě

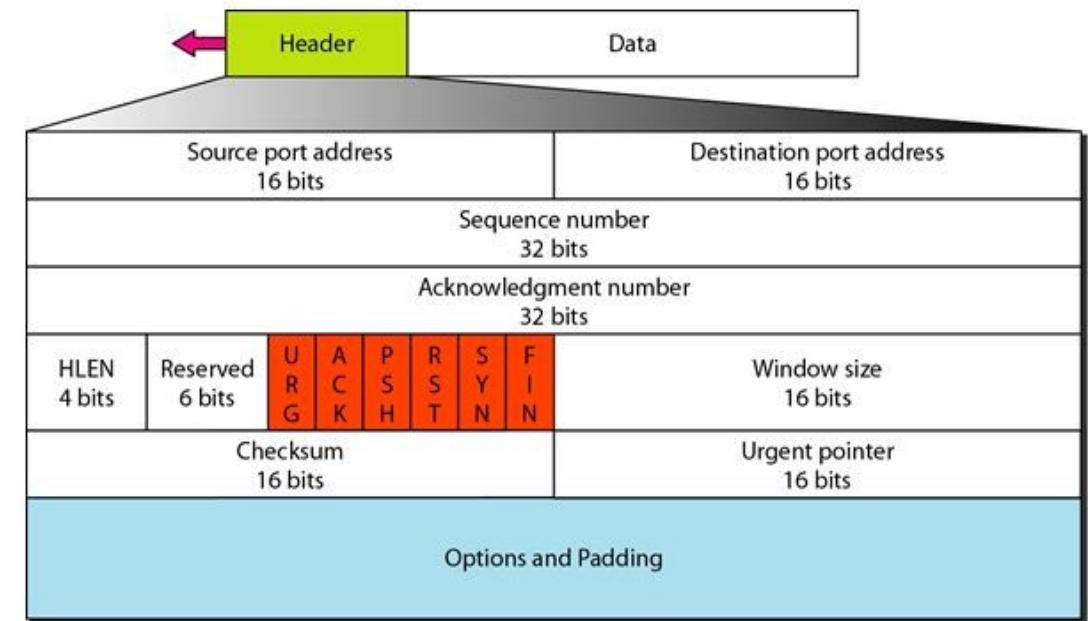


zdroj:

[https://en.wikibooks.org/wiki/Communication\\_Networks/TCP\\_and\\_UDP\\_Proocols/](https://en.wikibooks.org/wiki/Communication_Networks/TCP_and_UDP_Proocols/)

# Prokoly v L4: TCP

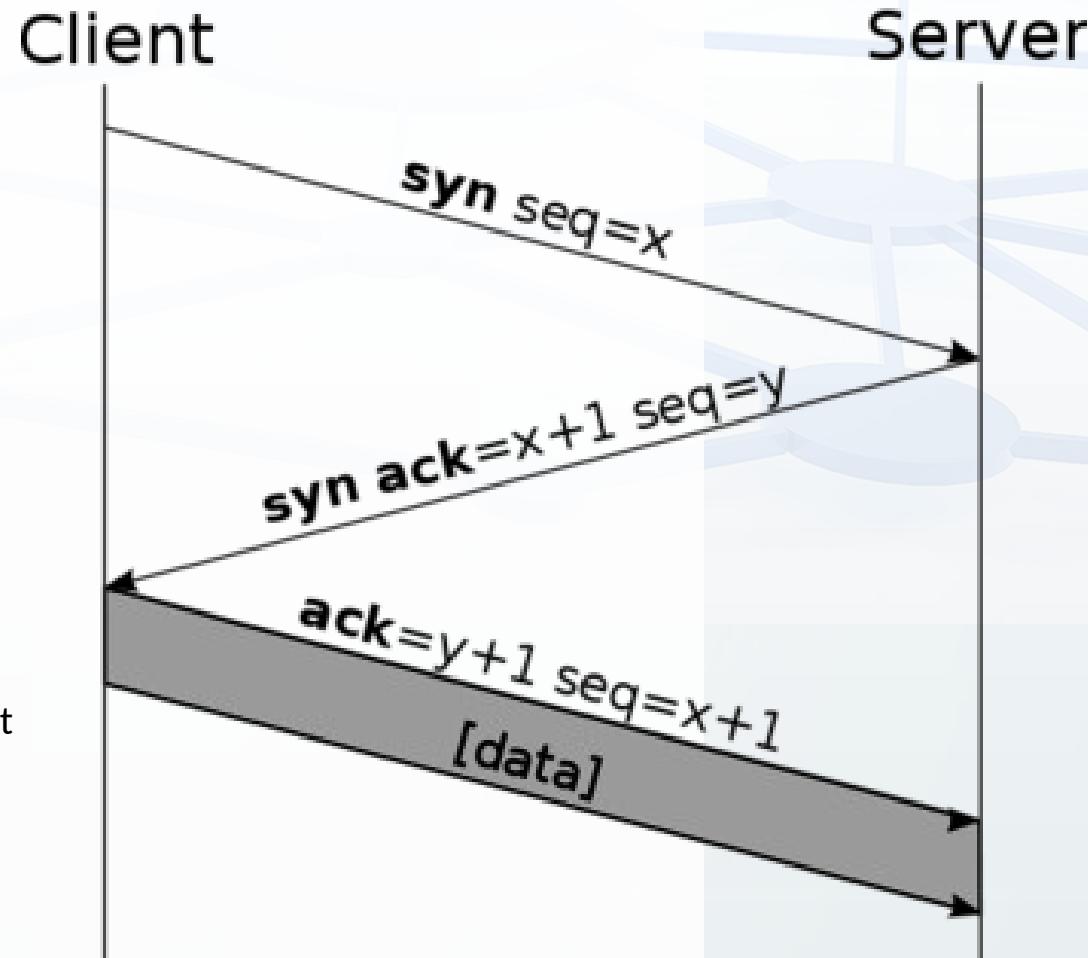
- TCP – Transmission Control Protocol
- Spojovaný
  - Před začátkem samotného přenosu je dohodnuto spojení
  - Příjemce tedy musí s přenosem souhlasit
- Potvrzovaný
  - Během přenosu je potvrzována správnost přenosu dat
    - Pokud během přenosu dojde k chybě jsou data znova vyžádána
- Používá bufferování data a dokáže přeskládat posloupnost
  - Pokud data dorazí v chybném pořadí, ale vejdou se do vstupního buf  
jsou v něm
    - správně seřazena – není vždy nutný re-transmit
  - Vyšším vrstvám poskytuje „zdání“ bytového streamu
    - Tedy že data tečou kontinuálně a nejsou dělena, ale to je jen iluze
- Podporuje řízení toku dat – používá protokol s klouzajícím okén
- Má vyšší režii než UDP
  - Kvůli navazovaní spojení, potvrzování zpráv, ukončování spojení atd
- Je spolehlivé, takže za vyšší vrstvy odvádí více práce



zdroj: <http://www.myreadingroom.co.in/notes-and-studymaterial/68-dcn/850-tcp-segment.html>

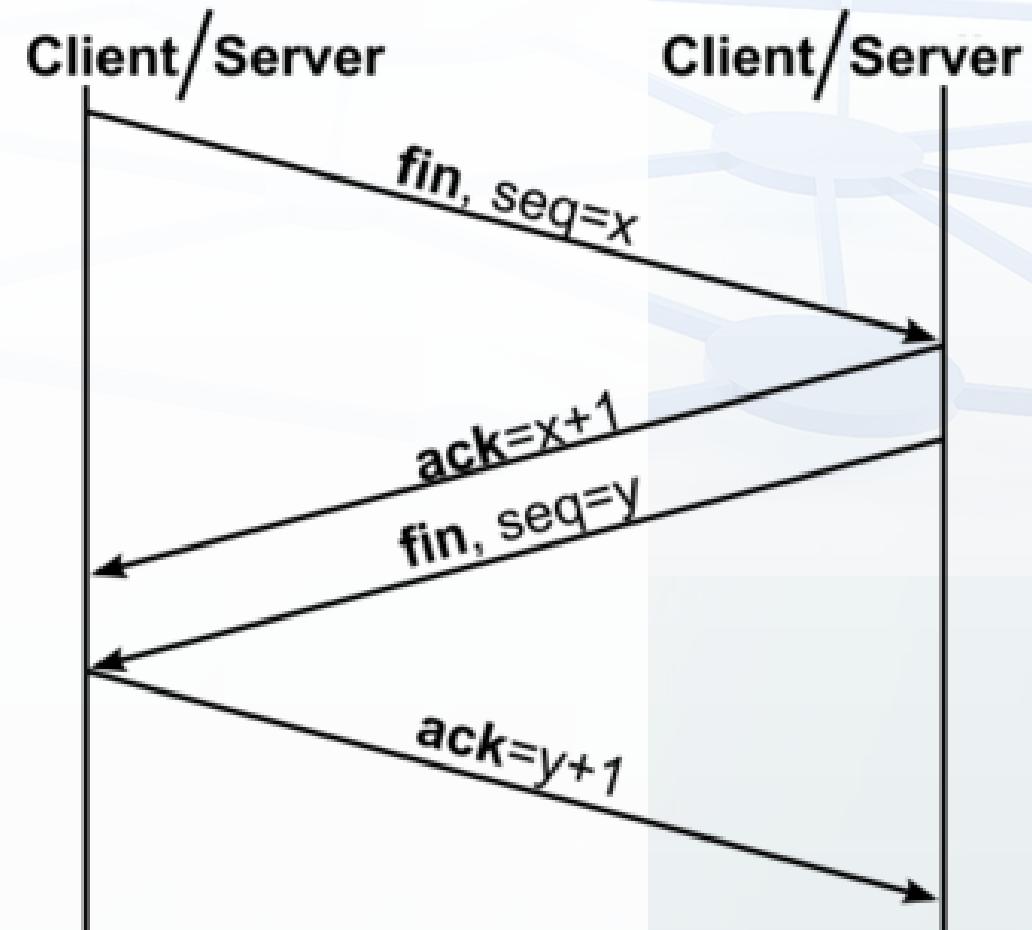
# Prokoly v L4: TCP: Navazování spojení

- Před každým přenosem musí být spojení „navázáno“
  - Proběhne „hand-shake“, čímž obě strany s přenosem souhlasí
- Pořadí operací:
  - 1) Klient posílán SYN se sériovým číslem segmentu X
    - říká „Ahoj“ chtěl bych se k tobě na daném portu a protokolu připojit
  - 2) Server – pokud souhlasí – odpovídá SYN + ACK
    - říká „Ok, jsem pro“ a zároveň potvrzuje přijetí zprávy X
  - 3) Klient odpovídá ACK a potvrzuje přijetí zprávy Y
    - říká „Ok jsme dohodnutí a může začít přenos“
- Samotných SYN zpráv se často využívalo k útoku DDOS
  - Klient posílal jen SYN, ale už na ně nereagoval
  - Na serveru se alokovalo spojení pro klienta a čekalo se na timeout
  - Spojení mohlo být je určité množství – dle ulimit počet otevřených spojení
  - Při vyčerpání došlo k tomu, že stará spojení fungovala, ale nová už nešlo navázat
  - Řešením je např v Linuxu zapnutí
    - echo 1 > /proc/sys/net/ipv4/tcp\_syncookies | sysctl -w net.ipv4.tcp\_syncookies=1
    - Spojení pak nejsou rovnou otevírána a nedochází tak k vyčerpání poolu



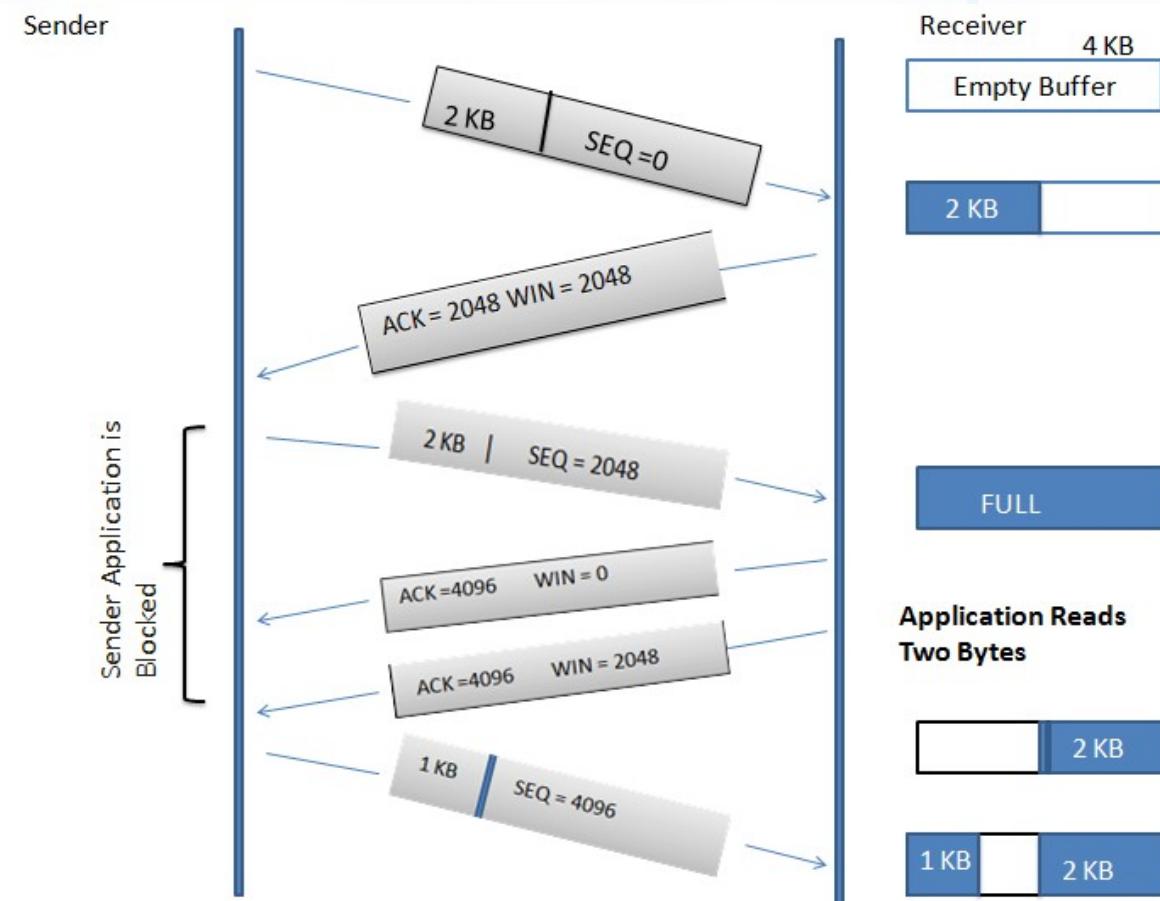
# Prokoly v L4: TCP: Ukončování spojení

- Tím, že je spojení v rámci TCP navazováno a udržováno, je nutné jej i korektně ukončit
  - Aby protistrana věděla, že spojení skončilo a přestala blokovat port+protokol
    - Velmi častý problém u semestrální práce, server spadne a 30s se nedá původní port a protokol použít
    - Logicky, protože k ukončení nedošlo korektně a server čeká na timeout, zda se klient ještě neozve
- Pořadí operací:
  - 1) Klient/Server posílán FIN se sériovým číslem segmentu X
    - říká „Chtěl bych ukončit naši komunikaci“
  - 2) Server/Klient – pokud souhlasí – odpovídá ACK
    - říká „Potvrzuji přijetí tvého požadavku“
  - 3) Server/Klient posílá FIN s pořadovým číslem Y (komunikace vyvolaná protistranou)
    - říká „Chci ukončit naše spojení“
  - 4) Klient/Server posílá ACK Y
    - potvrzuje doručení požadavku na ukončení spojení s číslem y a jelikož sám o ukončení žádal, nemá sním problém a spojení ukončuje
    - info o spojení mizí na obou stranách z alokačních tabulek a port+protokol je volný
- Pokud nedojde poslední ACK dochází k jednostrannému ukončení spojení
  - Definitivně jej vyřeší až timeout



# Prokoly v L4: TCP: Řízení toku dat

- TCP na rozdíl od UDP i od IP řeší řízení toku dat
  - A tím i předcházení zahlcení
- Používá se kontinuální kladné potvrzování
  - Můžeme odesílat další data dříve, něž dorazí potvrzení na odeslaná data
- Používá protokol s klouzajícím okénkem
  - Okénko definuje kolik dat může být odesláno bez potvrzení
- Přímce v odpovědi potvrzuje kolik dat v pořádku přijal a zároveň kolik je ještě schopen bezpečně přjmout
  - Pokud má klient plný buffer, posílá WIN=0
    - Další data nemohou být poslána – pokud by k odeslání došlo, budou velice pravděpodobně zahozena
  - Pokud se buffer uvolní posílá znova potvrzení posledních dat co má a zároveň velikost dat, které může přjmout



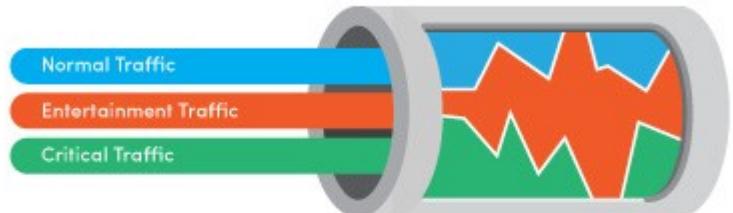
# Prokoly v L4: TCP: Předcházení zahlcení

- Zahlcení může nastat ze dvou důvodů
  - Příjemce nestihá přijímat / zpracovávat data
    - Pokud to jde ( stihne se ) je v rámci TCP řízen pomocí WIN – viz předchozí slide
    - Pokud to nejde ( nestihne se – dat přišlo tolik najednou, že už je nedokáže zpracovat ) jsou data zahozena a odesílatel čeká na timeout, ten mu signalizuje, že patrně došlo k zahlcení a odešla data znova, ale s nižším WIN
  - Příjemce stihá, ale úzkým hrdlem je síť
    - Pokud data ještě „nějak trochu“ tečou, kontroluje odesílatel RTT a dle něj určí ( při prodloužení ), že je v síti patrně problém a je třeba snížit WIN a tím zpomalit datový tok
    - Pokud se síť už zahltila, dojde na straně odesílatele k timeoutu, na základě kterého opět předpokládá problém v síti ( nebo na přijímači, to je v daný okamžik jedno ) a nově posílá data s menším WIN

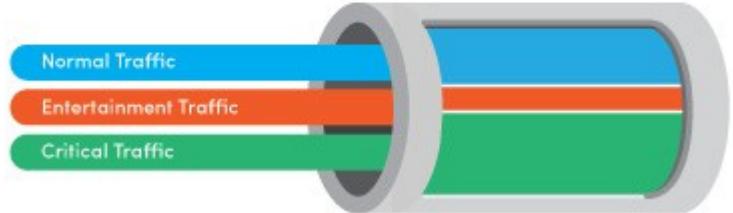
# QoS: Quality of Service

- Výchozí strategie přenosu je „best effort“
  - Ke všem se přistupuje stejně, nerozlišuje se, která služba je přenosem realizována
- Je to spravedlivé, ale v reálném provozu komplikované
  - Reálně nepotřebují všechny služby garanci parametrů přenosu
    - Například SMTP či HTTP se bez bez QoS v pohodě obejde
      - Ale i zde jde použít, pokud je to třeba – například garance části přenosového pásma pro přístup k firemnímu IS
    - Ale zároveň existují služby, kde je garance parametrů přenosu nutnost
      - Například VOIP
- Parametry, které můžeme chtít garantovat pomocí QoS:
  - Propustnost ( šířka pásma )
  - Maximální absolutní zpoždění
  - Maximální průměrné zpoždění
  - Chybovost přenosu
  - Rozptyl zpoždění ( jitter )

**Bandwidth WITHOUT QoS**



**Bandwidth WITH QoS**



# QoS: Možnosti řešení

- Zajištění správné funkce služeb vyžadující garanci některých přenosových parametrů jde zajistit více způsoby:
  - „Přístup hrubou silou“ - vlastně nic neměníme, ale výrazně předimenzujeme linku
    - Problém tam pořád je - může dojít k selhání – ale výrazně snížíme pravděpodobnost
    - V praxi často použité, protože se sice jedná o „dražší“, ale realizačně nejjednodušší cestu
      - Nemusím nic rekonfigurovat, jen zaplatím za lepší/garantovanou službu
  - Nasazení doplňkových opatření
    - Například „client buffering“
      - Problém v síti – nestabilita – pořád je, ale tím, že přijímám více dat do bufferu na klientovi, jsem schopen tento problém „skrýt“
      - Defakto se opět jedná o řešení „hrubou silou“ - protože více paměti není zdarma
      - Samozřejmě hodí se jen pro některé typy přenosů – např. Neinteraktivní video-audio přenos
  - Nahrazení „best effort“ pomocí QoS
    - Tedy zkusíme nějak zajistit rozdílný přístup k přenášeným datům na základě toho, o jakou službu se jedná

# QoS: Možnosti implementace

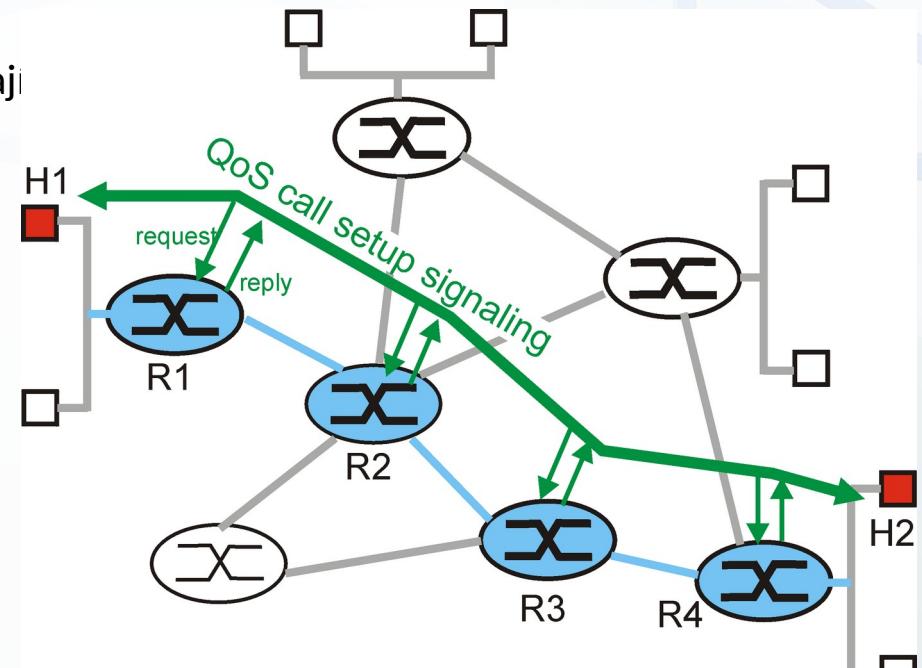
- QoS je možné řešit na více vrstvách ISO/OSI / TCP/IP
  - Konkrétně na L2, L3 a L4
  - L2 – podpora přímo v některých protokolech jako ATM či Frame Relay
  - L3 – pokud L3 používá „best effort“ musí být chování vhodně změněno a to na všech směrovačích v cestě
    - Logický předpoklad, protože pokud jen jeden router v cestě nebude toto respektovat, nemůže QoS nikdy fungovat
  - L4 – využití specifikace port+protokol – tedy rozlišení služeb
- V rámci TCP/IP nebyl původně QoS řešen
- Následně byl doplněn ve dvou variantách
  - DiffServ – Differentiated Services
    - Princip prioritizace
  - IntServ – Integrated Services
    - Princip garance

# QoS: Principle

- Značkování paketů
  - Jednotlivé paketu budou rozděleny do tříd dle požadovaných parametrů
  - Router pak pracuje s pakety různě na základě značky
  - Jednotlivé třídy musí mít různé priority
    - WWW přenos bude mít nižší prioritu než VOIP, protože web dokáže dočasně vybrat celé pásmo a blokovat hovory po VOIP
- Izolace jednotlivých tříd
  - Řazení tříd dle požadované šířky pásma
  - Nutná kontrola požadovaných parametrů – aby se neprekračovalo co je dohodnuto
- Efektivní využití šířky pásma
  - Nedovolovat realizovat přenosy, které překračují šířku pásma
  - Logický požadavek, protože ty budou vždy dělat problém
- Kontrola vstupu
  - Přenos musí před zahájením informovat jaké zdroje bude potřebovat
  - Pokud síť nedokáže takové požadavky zajistit, může přenos nerealizovat

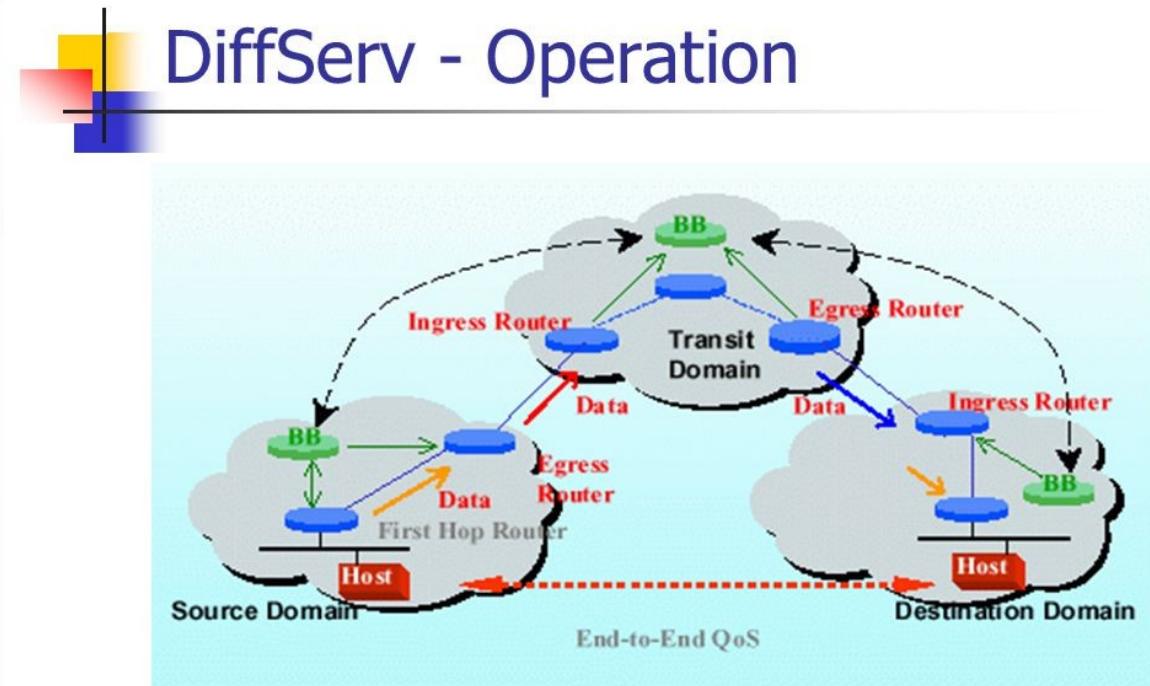
# QoS: Možnosti implementace: Integrated Services

- Integrated Services – jednotné služby
  - Funguje na principu rezervace zdrojů
    - Díky tomu garantuje požadované zdroje
  - Část zdrojů je obecnému IP odebrána a jejich využití / rozdělené řeší QoS
  - To realizuje alokace po celé cestě v síti
    - Defakto vytváří virtuální okruh, které se v přenosech dle požadovaných priorit střídají
  - V rámci L4 se realizuje tak, že při vytváření spojení se definují požadavky pro přenos a ty jsou pak požadovány po L3
  - Pokud požadavky není možné zajistit není přenos zahájen / realizován
    - Proč také, když předem víme, že nedopadne
  - Nutným požadavkem je existence možnosti odebrání zdrojů IP a rezervace pro QoS
    - To řeší RSVP – ReSerVation Protokol
      - Projde všechny routery v cestě a vyjedná vyčlenění požadovaných zdrojů pro QoS
      - Podporuje unicast i multicast
      - Používá existující směrovací tabulky – nemění směrování, řeší jen rezervaci



# QoS: Možnosti implementace: Differentiated Services

- Differentiated Services – rozšířené služby
- Definuje třídy služeb a k nim přiděluje priority
- Každý paket je klasifikován při vstupu
  - Neřeší se na každém směrovači, jen na hraničních směrovačích a v dané podsítí se klasifikace jen požije
    - Protože v jedné síti budou platit stejné pravidla nemá cenu data neustále překlasifikovávat
- Každý IP paket si sebou nese informaci o tom do jaké třídy patří
  - Klasifikace se přenáší v rámci TOS pole v IP hlavičce
- Jednotlivé směrovače nakládají s paketem dle uvedené třídy
  - Na základě priorit mohou uspíšit / zpomalit odbavení
- Je nutné, aby byl protokol podporován všemi směrovači v cestě
  - Pokud by nebylo splněno nemůže prioritizace fungovat
- Není řešeno centrálně v rámci celé cesty, ale v rámci jednotlivých směrovačů
  - Tedy každý rozhodne samostatně na základě třídy a její priority
  - Není nutné udržovat „cestu“ a její stav
- Nepřináší tak vysokou jistotu splnění požadavku jako Integrated services, ale umožňuje levněji realizovat rozlišení priorit

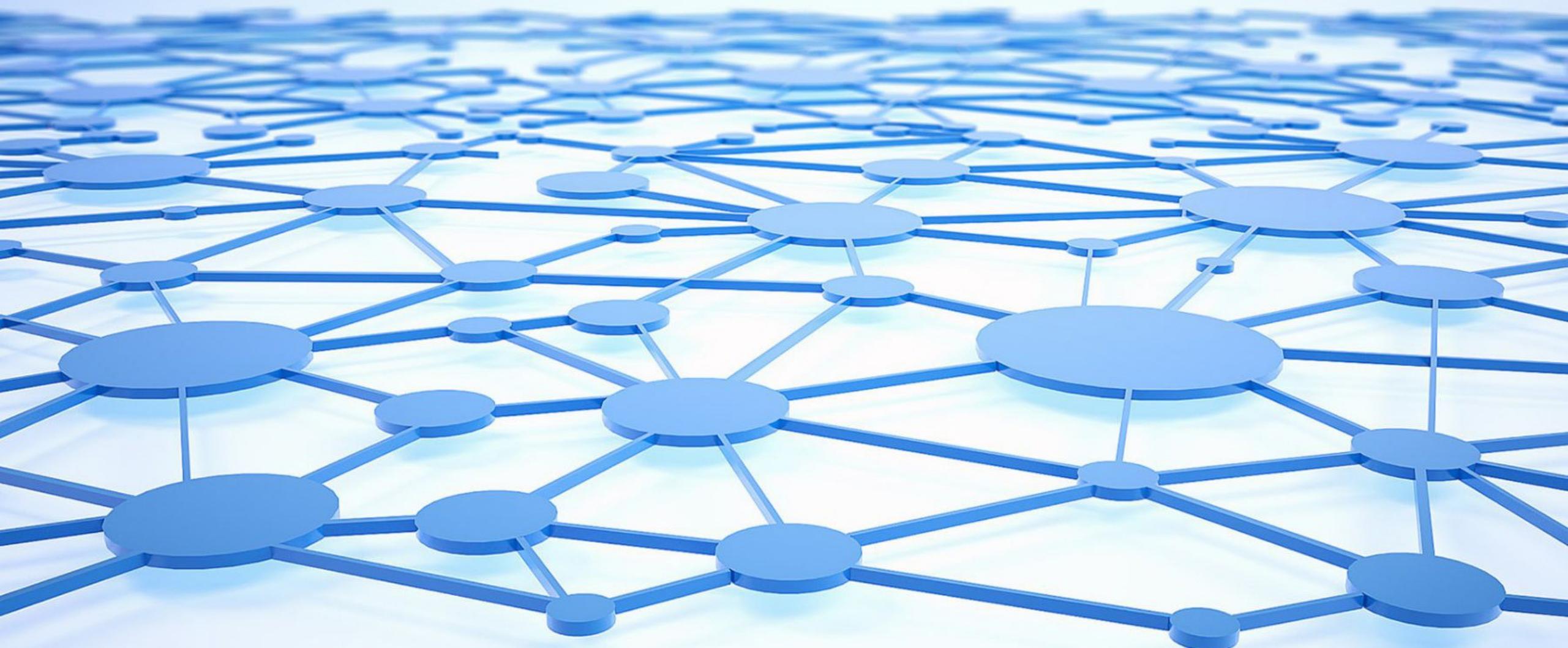


# QoS: Reálné použití

- L3 ani L4 není v provedení TCP/IP na QoS ideálně připraveno
- Nejčastěji se řeší posílením konektivity
- Kde se ale QoS řeší často jsou hraniční směrovače firemních / organizačních sítí
  - Tam je trochu jiná situace, protože tam definuji prioritu provozu na jednom místě
    - A tedy roztrídění do tříd
  - Cíl je stejný jako obecné QoS – zajistit důležitým / citlivým službám potřebné zdroje
    - VOIP /video hovory
    - Přístup na firemní systémy
    - VPN na další pobočky či k partnerům
  - Třídy mohou mít pevně danou šířku pásma
    - Jednodušší, ale méně efektivní varianta
  - Třídy mohou část zdrojů sdílet
    - Složitější na výpočet a udržení, ale dovoluje lepší využití zdrojů

# Úvod do počítačových sítí

Přednáška 11 ( 2025/2026 )  
ver. 2025-11-18-01

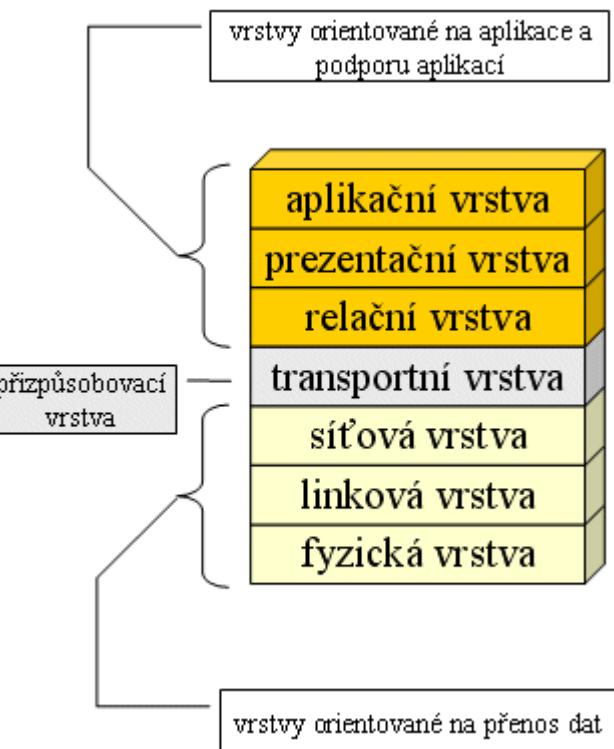


# L5 – Relační vrstva

# L6 – Prezentační vrstva

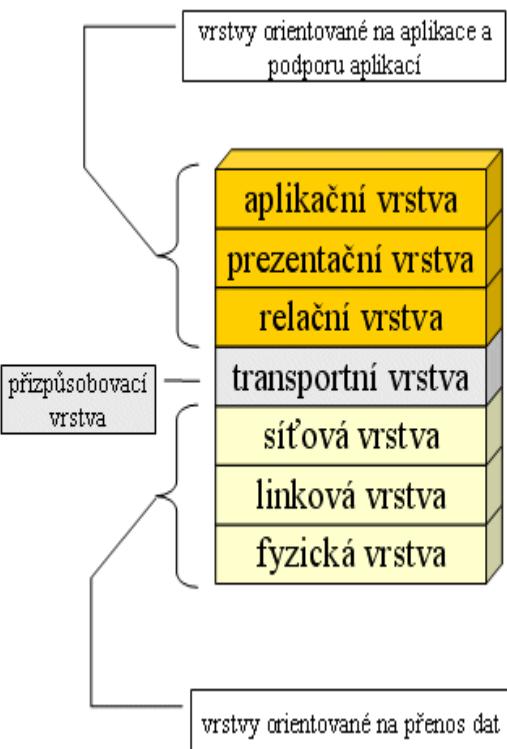
# L7 – Aplikační vrstva

- Vrstvy orientované na podporu aplikací
- L5 - Relační / session vrstva
  - Podpora služeb pro řízení relace – dialogu
  - Reálně dnes není implementována
- L6 – Prezentační vrstva
  - Prezentace dat – transformace dat z formátu, který potřebuje aplikace do formátu vhodného pro přenos sítí
  - Reálně dnes není implementována
- L7 – Aplikační vrstva
  - Vrstva umožňující síťové fungování aplikací
  - Postupem času výrazně zobecněná podoba



# L5 – Relační vrstva

- Relační / session vrstva
- Zajišťuje spolehlivý přenos mezi aplikacemi
  - Tedy může opět překrývat nedostatky nižší vrstev
- V rámci ISO/OSI byla vrstva navržena, ale reálně je dnes její funkce zajišťována až na aplikační úrovni
  - Tedy funkce této vrstvy – a prezentativní vrstvy také – jsou třeba a jsou požívány, ale jsou realizované v konkrétních aplikacích různě, nikoliv prostřednictvím služeb relační vrstvy
- Základní úkoly
  - Tvorba, udržování a ukončování relace – session
  - Řízení dialogu
  - Realizace synchronizačních bodů pro přenos
  - Řízení transakcí
- Příklady protokolů relační vrstvy
  - X.225, X.215
- Příklad protokolu, který relační vrstvy „nahrazuje“ v rámci TCP/IP může být například RPC



# L5 – Relační vrstva: Udržování relací

- Zajistit udržení relace pro dvě komunikující aplikace bez ohledu na reálné chování L4 vrstvy
  - Jedna relace v jednom L4 spojení
    - Nejjednodušší situace – navážu jedno spojení – a realizuji jeden HTTP request – například
      - Není nic před a nic po – relace si odpovídají 1:1 a tedy není „téměř“ třeba nic řešit
  - Jedna relace je realizována ve více L4 spojeních
    - Například v e-shopu košík – dáte položku do koše v rámci jednoho TCP spojení, ale pak odejdete od PC a až se vrátíte vaše TCP spojení je již díky timeoutu uzavřené, ale svůj košík stále chcete
      - Řešení pomocí session na úrovni aplikace / nebo cookie nebo kombinace, ale vše na aplikační vrstvě
  - Více relací prostřednictvím jednoho L4 spojení
    - Tedy zas v rámci e-shopu například možnost změnit identitu – odhlášení a přihlášení na jiného uživatele
      - Opět reálně realizované na L7 v rámci session / záznamech v DB

# L5 – Relační vrstva: Řízení dialogu

- Možnost zavádět dodatečná omezení komunikace
  - Dodatečná – realizovaná **NAD L4**
  - Tedy L4 komunikace může podporovat full-duplex, ale na L5 nemusí být podporován
- Realizace může být řešena formou „předávání pověření“
  - Podobně jako na L2 v Token Ringu - „Jen ten kdo má token – slovo – smí vysílat“
  - Reálně se samozřejmě předávání řeší jen pro polo-duplexní model
- Možné způsoby realizace
  - Simplexní
    - Komunikace vedená jen jedním směrem
  - Polo-duplexní
    - Zde je třeba řídit pověření – na úrovni L4 může být komunikace full-duplexní, ale charakter aplikace vyžaduje jeho omezení na polo-duplexní – princip „otázek a odpovědí“
      - V rámci HTTP posílám požadavek a čekám na odpověď – sice můžu poslat více požadavků paralelně, ale ty tvoří samostatné session
  - Full-duplexní
    - Komunikace je možná oběma směry bez omezení

# L5 – Relační vrstva: Synchronizace

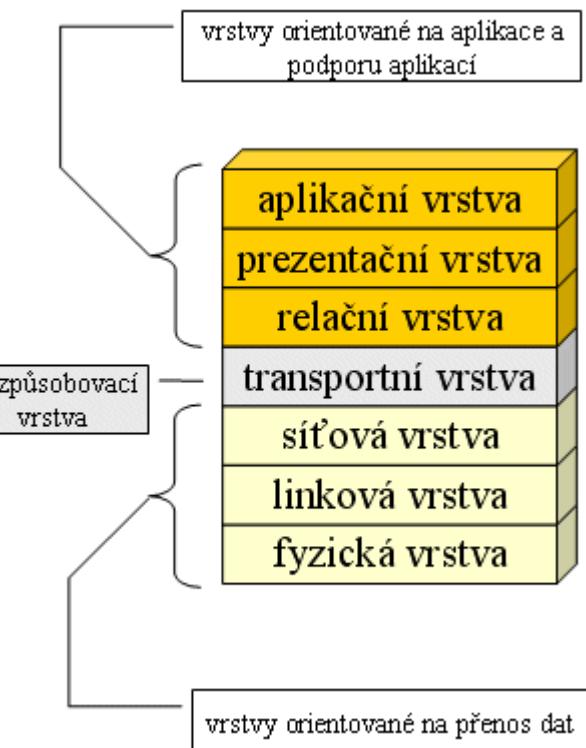
- Vytváření synchronizační body v komunikaci
- Snaží se řešit situaci, kdy jsou data na úrovni L4 v pořádku přenesena, ale v rámci aplikace pak nejsou v pořádku zpracována
  - Například všechna přijímaná data tisknu na tiskárně a dojde papír
- Aby bylo možné navázat na předchozí případ, zavádí se kontrolní synchronizační body – checkpoints
- Tyto body mohou být dvojího druhu
  - Hlavní synchronizační body – major - potvrzované
    - Definují bod v rámci komunikace, ke kterému už ale odesílatel nemá připravená data
    - Mezi hlavními body jsou vedlejší kontrolní body
  - Vedlejší kontrolní body – minor - nepotvrzované
    - Leží mezi dvěma hlavními body
    - V rámci komunikace je možné se k jednotlivým vedlejším kontrolním bodům vracet a odesílatel musí mít tato data připravena k znovu odeslání
  - Poslední hlavní kontrolní bod definuje oblast kam až se můžu v datech vrátit
  - Jednotlivé vedlejší kontrolní body až k poslednímu hlavnímu kontrolnímu bodu definují kroky po kterých se mohu vracet

# L5 – Relační vrstva: Řízení transakcí

- Relační vrstva zavádí
  - aktivity - činnost například přenos souborů
    - Činnost od připojení k ukončení spojení
    - Aktivita je ohraničena hlavními synchronizačními body
  - dialogová jednotka
    - Přenášená data – například přenos jednoho souboru je jedna dialogová jednotka
    - Dialogová jednotka může obsahovat více vedlejších synchronizačních bodů
- Jednotlivé aktivity jsou na sobě nezávislé
  - Zda proběhla předchozí aktivita nemá žádný vztah k následující aktivita
- Každá dialogová jednotka musí proběhnout celá nebo vůbec
  - Tedy se jedná o atomické operace

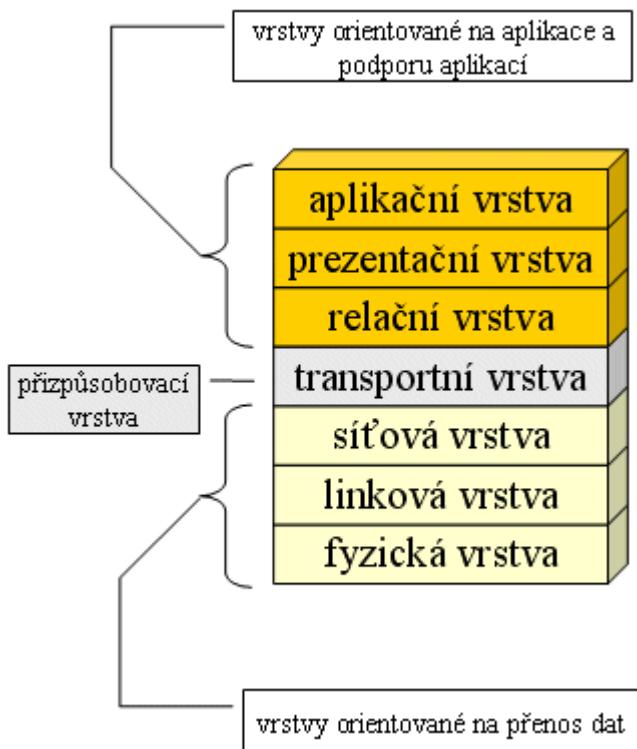
# L6 – Prezentační vrstva

- Zajišťuje konverzi aplikacích dat do podoby vhodné pro přenos počítačovou sítí
  - Univerzální formát – tedy vlastně popis jaká data přenáším a jak mohou vypadat
    - Můžeme mít dva přístupy:
      - „Jednofázová“ - data nepopisuji, ale pouze přenáším, ale z definice protokol vím, o jaká data jde a jak s nimi pracovat
      - XDR – External Data Representation - Přenáší jen data – musím vědět jaká data jsou přenášena
    - „Dvoufázové“ řešení
      - Abstraktní syntaxe
        - Popis dat / datových struktur
        - ASN.1 – Abstract Syntax Notation
      - Přenosová syntaxe
        - Jak bude vypadat ten samotný přenos
        - BER – Basic Encoding Rules
          - TLV – Type Length Value
            - Nemusím vědět jaká data přenáším, protože o jaké data se jedná je součástí přenášených dat
  - Komprese data
    - Snaha snížit velikost přenášených dat
  - Šifrování dat
    - Snaha zabezpečit data během přenosu
  - Pro každý řešený problém může existovat a být podporováno více variant
    - Na začátku musí proběhnout dohoda jak přenášet
    - Obě strany znají několik možností / protokolů a musí se dohodnout na nejvhodnější kombinaci
    - Dnes využíváno například pro šifrování SSL



# L7 – Aplikační vrstva

- V obou modelech je aplikační vrstva, ale má dramaticky jiný význam
  - OSI/OSI – jedná se o službu poskytovanou aplikacím
    - Společné služby
      - Vytvoření asociace – vzájemná komunikace – navázání spojení
      - Typy služeb
        - » Volání vzdálených podprogramů
        - » Transakční zpracování – atomické operace
        - » Spolehlivý přenos dat
    - Specifické služby
      - Konkrétní služby, které požadujeme
      - Např
        - » Přenos souborů( FTP, TFTP, ... )
        - » Adresárové služby ( Novell, MS AD, ... )
        - » Vzdálený přístup k terminálu ( RDP, SSH, Telnet, ... )
        - » Přenos zpráv( Email, Instant messaging )
  - TCP/IP – jedná o prostředí, kde opravdu běží samotná aplikace
    - Jednotlivé aplikace si výše uvedené problémy řeší samostatně
    - Můžeme definovat dva typy aplikací
      - Systémové – to co je třeba pro fungování sítě
        - » DNS, LDAP, DHCP, SNMP, ...
      - Uživatelské – to co požaduje uživatel
        - » FTP, HTTP, SMTP, SSH, ...



# L7 - Aplikační vrstva:Systémové aplikace

- Může dále dělit dle použití
  - Adresářové služby
    - Specifické typy databází
    - DNS, LDAP, „MS AD“
  - Konfigurační služby
    - BOOTP, DHCP
  - Služby síťového managementu
    - SNMP, RMON
  - Bezpečnostní služby
    - Kerberos, SASL, „MS AD“

# L7 - Aplikační vrstva:Systémové aplikace: DNS

- DNS – Domain Name System
- Převod jména na IP a zpět
  - proteus.fav.zcu.cz < - > 147.228.63.11
- Databáze s více typy záznamů
  - A, AAAA, CNAME, MX, TXT, ...
- Přenáší data nešifrovaně
- Používá TCP i UDP protokol a ve výchozím stavu port 53
- Odpověď může být cachovaná
- Odpověď může být
  - autoritativní
    - Od serverů uvedených v WHOIS DB – autoritativní jmenné servery
  - Neautoritativní
    - Od cachujícího serveru

# L7 - Aplikační vrstva:Systémové aplikace: DNS II.

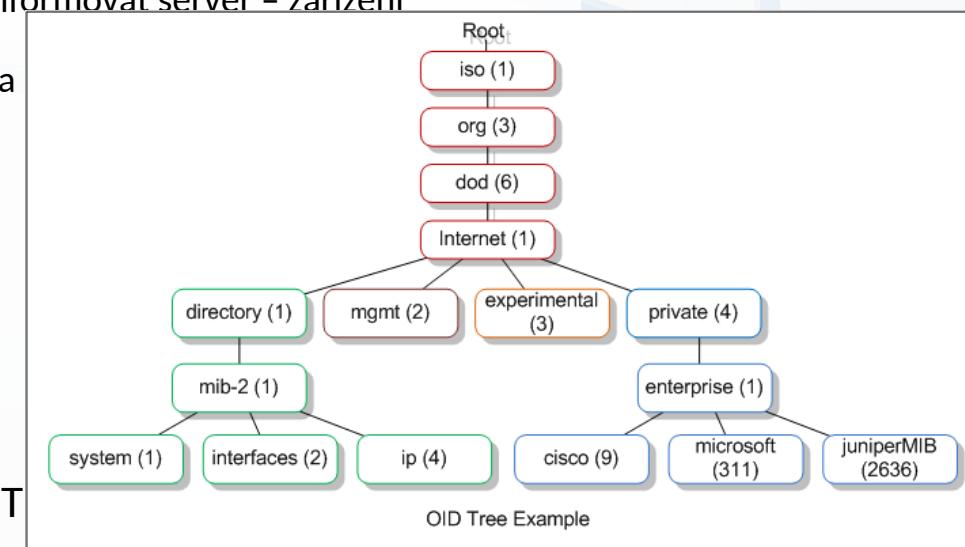
- Typické využití TCP a UDP v DNS
  - UDP – klasické komunikace – dotaz odpověď
    - Předpokládáme malá data – nečekáme, že dojde k chybě
    - Těch dotazů může být hodně a snadno se mohou opakovat
  - TCP – typicky synchronizace zón
    - U přenosu zónových souborů se jedná o větší celky dat
    - Tento přenos je třeba aby byl spolehlivý a nejde zde o čas jak rychle se povede
      - Rozuměj, vteřiny nehrají roli
- DNS není v základu nijak zabezpečené a je možné poměrně snadno
  - Podvrhnout odpověď – pokud jsem na vhodném místě mohu odpovídat a jiný DNS server a tím přesměrovat provoz
  - Podvrhnout dotaz – jedná se o UDP, tedy datagramy, které nemají navazované spojení, pokud vytvořím falešný – s falešným odesílatelem – DNS servery na něj odpoví, ale NE mě, ale podvrženému odesílateli a tím dojde na základě „zrcadla“ k DDOS
- Zabezpečení DNS se řeší až s příchodem DNSSEC a podepisováním zónových záznamů
  - Zatím není povinné a tedy ani masivně rozšířené

# L7 - Aplikační vrstva:Systémové aplikace: DHCP

- Protokol sloužící k auto-konfiguraci nastavení sítě
- Používá UDP protokol dva porty
  - 67/UDP – port na kterém se klienta ptá serveru
    - Respektive na začátku se ptá všech broadcastem – protože žádné nastavení sítě nemá
    - Tím že nastavení sítě nemá, jedná se L2 broadcast
    - Zdrojová MAC je nastavena na MAC klienta
    - Zdrojová IP je 0.0.0.0 a cílová je 255.255.255.255
  - 68/UDP – port na kterém server odpovídá
    - Odpověď jde opět broadcastem( ale může i s cílovou MAC pro klienta ) se zdrojovou MAC adresou serveru, IP zdrojová je server, cílová je 255.255.255.255 ( dokud není konfigurace hotová, klient IP nemá )
- Provoz, tím že se používá broadcast je omezen na LAN
- Pokud v dané LAN není DHCP server, je možné požadavek „předat“ dále do jiné sítě, pokud je routeru DHCP Realy Agent
  - Mimo vlastní síť jde požadavek běžně unicastově – protože může projít pře více LAN
- V síti by měl být pouze 1 DHCP server jinak nastává problém
  - Ono jich tedy může být více, ale musí tvořit cluster – spolupracovat jako fail-over / balancer
  - Pokud je jich více síť nemusí fungovat správně
    - Od koho adresu mám jde najít v logu nebo tcpdumpu
    - Servery mohou mít i stejnou IP – pak je můžeme rozlišit pomocí ARP - MAC

# L7 - Aplikační vrstva:Systémové aplikace: SNMP

- SNMP - Simple Network Management Protocol
  - Protokol sloužící pro podporu správy sítě a sběru dat
- Používá více portů
  - Nezabezpečená komunikace agenta UDP 161
    - Například klasické dotazy agenta - „Kolik dat proteklo přes interface eth0?“
  - Asynchronní nezabezpečené trapy UDP 162
    - Pokud v síti nastane nějaká monitorovaná událost, umí o tom SNMP asynchronně informovat server – zařízení kde problém nastal kontaktuje server
    - Velice výhodné tam, kde máme málo událostí – šetříme linku – a například adresy za „nemůžeme“ jiné monitoringy jako např Icinga2
  - Zabezpečená komunikace agenta UDP 10161 – dostupné až od verze SNMPv3
  - Zabezpečené asynchronní trapy UDP 10162 – dostupné až od verze SNMPv3
- Informace shromažďuje ve vlastní konfigurovatelné databázi MIB
  - Nemá pevný obsah, ale definuje identifikátory OID a k nim hodnoty
    - Např. 1.3.6.1.2.1.2.2.1.5
  - Využívá k popisu dat podmnožinu ASN.1
    - Tedy „jako by“ prezentační vrstvu
- Nejčastěji se používá jako podkladová platforma pro sběr nejrůznějších dat o IT
  - Protože ve velkém množství zařízení je přímo podporován
- Sice k němu existuje „modernější“ nástupce – RMON, ale ten se zatím masivně neprosadil



zdroj: <https://www.fi.muni.cz/~kas/pv090/referaty/2015-podzim/snmp.html>

# L7 - Aplikační vrstva: Uživatelské aplikace

- Může dále dělit dle použití
  - Přenos souborů
    - TFTP, FTP, SCP, Rsync, HTTP
  - Vzdálený přístup
    - Telnet, X-window, SSH, RDP, VNC
  - Přenos zpráv
    - SMTP/POP/IMAP, Skype, Jabber, ...

# L7 - Aplikační vrstva: Uživatelské aplikace: TFTP

- TFTP - Trivial File Transfer Protocol
- Používá UDP port 69
- Nepodporuje žádné zabezpečení či ověření
- Nepodporuje žádné složitější operace než download a upload souboru
  - To je ovšem na jednu stranu výhodné, protože je možné jej integrovat do HW či použít na velice nízké úrovni v OS, protože je jednoduchý
- Typicky se používá na
  - přenos základní konfigurace routeru/switche
  - Boot bezdiskových stanic v kombinaci s DHCP a např NFS
    - Přes DHCP zjistíme nastavení sítě a info, odkud máme stáhnout kernel – TFTP a odkud máme připojit rootFS – NFS
- Díky omezeným možnostem se příliš nehodí na větší a časté přenosy

# L7 - Aplikační vrstva: Uživatelské aplikace: FTP

- FTP - File Transfer Protokol
- Slouží pro přenos souborů - filozoficky se jedná o zdokonalení TFTP
- Používá TCP protokol
  - Protože potřebuje mít jistotu správného přenosu
- Používá více TCP portů – v základu 20/TCP a 21/TCP a další náhodně zvolené dle zvoleného módu
- Kromě prostého přenosu souborů umožňuje i další operace
  - Přenos může být binární nebo textový
  - Je možné procházet a vylistovávat adresáře
  - Je dostupné podpora ověření jménem a hesel
    - Ale je zachována i možnost anonymního přístupu
  - Je dostupné podpora šifrování pomocí SSL
    - Jen v novějších verzích – v základní je jméno a heslo posíláno v plain podobě
    - SSL verze nepoužívá extra port, ale je v rámci řídícího portu 21 dohodnuto, zda je SSL podporované a požadované oběma stranami
  - Jsou dostupné dva operační módy
    - Aktivní/normální mód
    - Pasivní mód

# L7 - Aplikační vrstva: Uživatelské aplikace: FTP: Aktivní mód

- Aktivní / normální móde
  - Jedná se o výchozí chování
  - Klient se portu 21 připojí k serveru provede ověření atd
  - Před přenosem zvolí klient port na kterém začne poslouchat / očekávat spojení od serveru
    - Tento port je na server přenesen v rámci session na portu 21 a příkazem PORT
  - Server se připojí k klientovi na zvoleném portu s odchozím port 20 na straně serveru
  - Problém nastává v případě, že je klientem za NATem
    - Klient na privátní IP sice poslouchá na portu, ale server se k němu nemůže pře veřejnou IP připojit, protože v NAT není adekvátní záznam pro nový port
    - Lze řešit pomocí speciálních modulů pro NAT, které poslouchají provoz na porty 21 a dovolují doplnit NAT tabulky pro realizaci přenosu
    - Problém nastává ve chvíli, kdy je provoz na jiném portu nebo kdy je provoz tunelován
      - Protože pak přídavný modul nemá jak zjistit potřebné parametry
- Problém může nastat i v případě použití state/established connection firewallu na serveru
  - Ten blokuje veškeré odchozí spojení, kterému nepředchází požadavek a navázání komunikace ze strany klienta
  - Cílem je blokování backdooru, ale zde se vlastně otevře náhodný port na který může kdokoliv přistoupit

# L7 - Aplikační vrstva: Uživatelské aplikace: FTP: Pasivní mód

- Pasivní móde
  - Je řešením problému aktivního módu a NATu
  - Klient se připojí k serveru na portu 21 a provede ověření atd.
    - A přepne komunikaci do pasivního módu příkazem pasv
  - Server zvolí náhodný port na kterém začne očekávat přenos
    - Zvolen typicky z dynamického rozsahu portů
  - Klient se druhým spojením připojí k serveru na dohodnutý port s odchozím portem 20
  - Zde problém s NATem nenastává, protože veškerá komunikace je inicializována ze strany klienta
  - Do pasivního módu je nutné se přepnout ještě před přenosem samotným i jinými příkazy
    - Protože i například vylistování adresáře je realizováno mimo port 21 a tedy by bylo blokováno

# L7 - Aplikační vrstva: Uživatelské aplikace: Telnet / SSH

- Protokoly pro vzdálené připojení na terminál ( ale ne jen to )
- Telnet starší a nešifrovaný, TCP port 23
  - Přesto, že jej dnes SSH nahradilo, stále se používá tam, kde SSH není podporováno, například switche či routery
    - Tím, že není šifrovaný je jednodušší na implementaci
  - Jelikož se jedná o protokol založený na přenosu text, je dnes hojně využíván k testování odolně orientovaných protokolů
    - Například HTTP, ale především POP a IMAP
- SSH modernější a nativně šifrované
  - Používá TCP port 22
  - Přenos může být nejen šifrovaný, ale i komprimovaný
  - Podporuje různé druhy autentizace
    - Jméno a heslo
    - Klíče např RSA
    - Externí ověření ala Kerberos
  - Umožňuje „tunelovat“ další protokoly
    - ssh [root@muj\\_server](mailto:root@muj_server) -L 3307:localhost:3306
      - Na localhost:3307 vám bude „poslouchat“ mysql ze server muj\_server, ale reálně bude spojení tunelované pomocí SSH

# L7 - Aplikační vrstva: Uživatelské aplikace: SCP / Rsync

- SCP umožňuje přenos souborů prostřednictvím SSH spojení
  - Stejně jako SSH používá TCP port 22
  - Na rozdíl od FTP podporuje od počátku šifrování celé komunikace – ne jen přihlášení
  - Funguje ve dvou režimech
    - SCP - dávkové zpracování příkazů – umí jen download/upload souborů
    - SFTP – Secure FTP- umožňuje interaktivní zpracování požadavků, stejně jako FTP klient
  - Podporuje chroot
    - Tedy při zadání cd / v klientovi neskončíte v rootu serveru, ale v přeneseném root, který je jícký pro každého klienta
- Rsync je program, který pomocí SSH dokáže přenášet soubory
  - Tedy obdobná funkce jako FTP/SCP, ale „chytrějším způsobem“
  - Rsync umí přenášet jen rozdíly nebo jen zobrazit jaká data být přenášel
  - Rsync umí navázat na předchozí přenos

# L7 - Aplikační vrstva: Uživatelské aplikace: VNC

- VNC – protokol sloužící k zobrazení vzdáleného grafického terminálu
  - Nativně používá port TCP 5900
  - Ověření probíhá na základě sdíleného hesla
    - Ale to není povinné, je možný i neautentizovaný přístup
  - Protokol není v základu nijak šifrován
    - To je možné obejít například použitím SSH tunelu/VPN či speciálních rozšíření
  - Provoz není komprimován a přenáší se celé obrazovky
    - Velice náročný na přenos, především tam, kde je změn hodně
    - Velice často má problém se synchronizací myši
      - Dat/změn je hodně a musí kromě stahování změn přenášet zpět na server pozici myši, což se děje se zpožděním – myš pak „ujízdí“
- V zásadě funguje jako „vzdálený pohled na obrazovku“, tedy je společné pro všechny uživatele
  - Na rozdíl od RDP, které vytváří separátní session pro každého uživatele
- VNC se dnes samo o sobě na vzdálený přístup příliš nepoužívá a nahrazováno modernějším RDP
- Ale stále je používáno jako součást dalších systémů
  - Například pro vzdálený přístup na konzolu virtualizovaných systému, kde požadován grafický interface – Microsoft Windows / MACOS

# L7 – Aplikační vrstva: Uživatelské aplikace: RDP

- RDP – Remote Desktop protokol - „Vzdálená plocha“ – protokol sloužící k zobrazení vzdáleného grafického terminálu
  - Nahrazuje VNC a řeší jeho nedostatky
    - Spojení je šifrované
    - Podporuje komprese
    - Přihlášení je ověřené na serveru na kterém služba běží
      - Tedy nemá své extra jméno a heslo jako u VNC, ale reálně dojde k přihlášení uživatele / vytvoření session/relace, na kterou je možné i navázat
      - Session může být na serveru i více pro různé uživatele
  - Ve výchozím stavu funguje na TCP portu 3389
  - Původně bylo jen součástí Microsoft Windows Serverů
  - Dnes dostupné na všech Microsoft Windows, ale i v dalších systémech jako je např Linux
    - A to jak klient – rdesktop / xfreerdp tak i server xrdp
  - Nepřenáší kompletní obrazovku, ale jen změny
    - Výrazně úspornější než VNC

# L7 – Aplikační vrstva: Uživatelské aplikace: Emailové služby

- Emailové služby jsou složeny z více komponent
  - MUA – Mail User Agent – například Outlook nebo Thunderbird
    - Slouží k obsluze pošty na straně klienta, tedy stahování, zpracování a tvorbu emailů
    - Používá více protokolů – POP3/POP3s, IMAP/IMAPS, SMTP/SMTPLS, HTTP/HTTPS, MAPI, ...
  - MTA – Mail Transfer Agent
    - Slouží k přenosu emailu od klienta k serveru a mezi servery samotnými
    - Používá protokol SMTP/SMTPLS
  - MDA – Mail Delivery Agent
    - Slouží k doručení email od MTA do schránky uživatele
    - Může se jednat o komunikaci pomocí pipe nebo pomocí LMTP protokolu

# L7 - Aplikační vrstva: Uživatelské aplikace: Emailové služby: POP3/POP3S

- Nejstarší protokol na stahování pošty
  - Neumí pracovat se složkami na serveru
  - Stahuje celé zprávy
  - Umí nechat na serveru kopie zpráv X dnů zpět
- Používat může dva porty
  - TCP 110 - pro nešifrované spojení
  - TCP 995 - pro šifrované spojení
    - To je realizováno na základě SSL certifikátu – o těch si řekneme více u HTTPS
- Komunikace je založena na principu přenosu textu, takže v nešifrované podobě je možné přímo vidět jméno a heslo
- Tím, že POP je textový protokol, je k jeho ověření/ladění možné použít telnet nebo openssl
  - Nešifrovaná verze:
    - telnet localhost 110
    - login pepa
    - pass pepa
  - Šifrovaná verze :
    - openssl s\_client -connect localhost:995
    - Příkazy jsou pak stejné, protože i protokol je stejný, jen je zabalený do SSL

# L7 - Aplikační vrstva: Uživatelské aplikace: Emailové služby: IMAP/IMAPS

- Modernější náhrada POP protokolu
  - Podporuje složky na serveru – včetně vnořených
  - Podporuje stahování pouze hlaviček emailu
  - Dovoluje přesouvat/filtrovat zprávy na serveru bez nutnosti stažení zprávy
  - Dokáže synchronizovat obsah mezi více klienty
- Používat může více portů
  - TCP 143 či TCP 220 - pro „nešifrované“ spojení
    - „nešifrované“ protože už umožňuje verzi STARTTLS a tedy, že i na portu 143 může být provoz šifrován, pokud se klient a server dohodnou, že je to požadováno a podporováno oběma stranami
  - TCP 993 - pro šifrované spojení
    - To je realizováno na základě SSL certifikátu – o těch si řekneme více u HTTPS
- Vnitřní komunikace je stejně jako u POP textová a tedy dovoluje ověření pomocí telnet/openssl
  - Nešifrovaná verze:
    - telnet localhost 143
    - . login pepa password pepa
  - Šifrovaná verze :
    - openssl s\_client -connect localhost:993
    - Příkazy jsou opět stejné, protože i protokol je stejný, jen je zabalený do SSL

# L7 - Aplikační vrstva: Uživatelské aplikace: Emailové služby: SMTP/SMTPS

- Protokol sloužící k přenosu emailu od klienta na server a následně mezi servery
- Používat může více portů
  - TCP 25 - pro „nešifrované“ spojení
    - „nešifrované“ protože už umožňuje STARTTLS a tedy, že i na portu 25 může být provoz šifrován, pokud se klient a server dohodnou, že je to požadováno a podporováno oběma stranami
  - TCP 465 - pro šifrované spojení
    - To je realizováno na základě SSL certifikátu – o těch si řekneme více u HTTPS
  - TCP 587 - pro šifrované spojení s podporou autentizace
    - Šifrování je řešeno pomocí STARTTLS
  - Dnes už je možné jak autentizaci tak šifrování provozovat i na portu 25
    - Ale používají se o zbylé dva a to jednak kvůli zpětné kompatibilitě, druhak proto, že port 25 bývá často při odchodu provozu z ISP blokován
- Stejně jako pop/imap je i SMTP textově orientován, takže základní připojení možné opět ověřit pomocí telnet/openssl
  - Nešifrovaně: telnet muj\_server 25
  - Šifrovaně: openssl s\_client -connect muj\_server:25

# L7 – Aplikační vrstva: Uživatelské aplikace: HTTP/HTTPS

- Hyper Text Transfer Protokol
- Běžně funguje na dvou portech TCP
  - http – nešifrovaná verze port 80
  - https – šifrovaná verze port 443
    - Jedná se o HTTP, který je „obalen“ šifrováním, ale uvnitř je stejný
- Původní verze přijala dotaz, poslala odpověď + návratový kód a spojení ukončila
  - To se v situaci kdy jedna HTML stránka má X částí jak JS, CSS, obrázky, atd ukázalo jako extrémně nevýhodné
    - Protože režie navázání spojení, forku procesu na serveru atd je drahá – dlouhá
  - Nyní podporuje keep-alive spojení – spojení zůstává otevřeno definovaný čas i po vyřízení požadavku, protože se očekává, že přijdou další – šetříme zdroje
- HTTP je bezstavový, stavovost – relaci – přináší až podpora cookie / session
  - Session může fungovat i bez podpory cookie, ale není o to obvyklé
  - Cookie představují bezpečnostní riziko – proto se dnes musí „potvrzovat“, že o nich víte a souhlasíte

# L7 – Aplikační vrstva: Uživatelské aplikace: HTTP/HTTPS – typy požadavků

- GET
  - Nejběžnější požadavek – dej mi soubor / stránku / data
  - Celá požadovaná URL např /login.php?username=pepa&heslo=pepa se zaznamenává do logu
    - Proto se nepoužívá k přenosu formulářů či k přihlašování
- POST
  - Běžně používaný k přenosu dat od klienta k serveru
    - Může jít o formulář s daty, ale i soubor či soubory
  - Do logů se propisuje jen URL, např /login.php, ale předávaná data už ne
- PUT / DELETE
  - vytvoření/smazání objektu na serveru
- HEAD
  - Stejně jako GET, ale vrací jen hlavičku odpovědi a ne data
  - Používá se k testovacím účelům

# L7 – Aplikační vrstva: Uživatelské aplikace: HTTP/HTTPS – návratové kódy

- Každá odpověď obsahuje informaci o stavu vyřízení požadavku
  - Odpověď v rámci L7 - níže může být více chyb, které jsou pro HTTP díky TCP transparentní
- Návratové kódy jsou členěné „po stovkách“ do skupin
  - 1xx - informativní
  - 2xx - úspěšné
    - Nejčastěji 200 OK
  - 3xx - přesměrování
    - Nebyla vrácena požadovaná odpověď, ale požadavek byl přesměrován na novou adresu, kde snad odpověď najdeme
  - 4xx - chyba vyvolaná / způsobená klientem
    - Přesněji chyba vzniklá v důsledky chování klienta, např
      - 401 - přístup na stránku, která vyžaduje autentizaci
      - 403 - požadavek na soubor, kterou server nemůže otevřít kvůli oprávnění na FS
      - 404 - požadavek na neexistující stránku
  - 5xx - interní chyba serverů

# L7 - Aplikační vrstva: Uživatelské aplikace: HTTP/HTTPS – zabezpečení

- HTTP protokol nepodporuje šifrování
- To přichází až s rozšířením na HTTPS
- Šifrování zajišťují SSL certifikáty
  - Před samotným přenosem dat je vytvořeno šifrované spojení a tím je pak přenášen HTTP protokol
  - Certifikáty mohou být
    - Vlastní nepodepsané – každý si jej pomocí openssl může vystavit sám
      - Problém je, že prohlížeč mu typicky nevěří
    - Podepsané vlastní certifikační autoritou
      - Pomocí openssl si vytvoříte i certifikační autoritu a tu nainportujete do prohlížeče
      - Váš prohlížeč už certifikátu věří, ale jen váš
    - Podepsané obecné známou certifikační autoritou
      - Placené – např Thawte, kupují se na 1 rok, pak je třeba jej obnovit
      - Zdarma – iniciativa Let's Encrypt – vystavují se zdarma na kratší dobu – např 3 měsíce
- Certifikát - „složité heslo s přidanou hodnotou“ je možné použít jak k šifrování tak i k ověření identity proti straně
  - Pokud certifikát podepsala nějaká obecně uznávaná autorita, byl nejprve jeho žadatel prověřen, takže pak máte vyšší jistotu, že se opravdu bavíte avizovanou protistranou
    - Samozřejmě i to lze za určitých okolností – například pomocí AD wildcard certifikátu zneužít

# L7 – Aplikační vrstva: Uživatelské aplikace: HTTP/HTTPS – Proxy/cache

- Téměř každý HTTP klient – browser – umí cachovat požadavky
  - Cílem je snížit opakovaný přenos dat a zrychlit odezvu webu
  - Vzniká zde problém neplatného/zastaralého obsahu cache
- HTTP požadavek nemusí jít na server přímo, ale může jít přes HTTP Proxy
- „běžná“ proxy funguje podobně jako cache na klientovi, ale jednotná pro celou síť
  - Opět s cílem snížit duplicitní přenosy, ale zde pro celou síť
  - S možností filtrace požadavků – např \\*porno\\* je zakázaný požadavek
  - S možností autentizace/autorizace požadavků – např na facebook.com se může, ale jen pokud znám jméno a heslo
    - Myšleno jméno a heslo k přístupu přes proxy server – následně se ještě logujete na facebook.com
  - Může zde docházet i k filtraci na úrovni IP adresy
    - Tedy vybrané IP – VLAN pana ředitele – smí na NET bez omezení, jiná VLAN má část obsahu zakázaný
    - Defakto se jedná o „firewall“, ale na L7 vrstvě
      - » Na L7 znamená, že nutně musím rozumět danému protokolu
- Proxy může logovat veškerá požadavky a používá se tak často jako podklad pro report provozu v rámci organizace
- Typickým představitelem proxy je například Squid

# L7 – Aplikační vrstva: Uživatelské aplikace: HTTP/HTTPS – Reverzní proxy

- Druhým typem proxy je reverzní proxy
  - Běžná proxy zaštiťuje organizaci a propouští provoz dále do internetu
  - Reverzní proxy je typicky na straně serverů a zprostředkovává komunikaci klienta s jedním či více vybranými servery
  - Cíle nasazení jsou částečně shodné s klasickou proxy
    - Cachování požadavků – proxy může například statická data jako JS/CSS odbavovat výrazně rychleji než server za ní
    - Filtrace požadavků na základě URL
    - Přepis požadavků – tedy změna části nebo celé URL či přesměrování
  - Zároveň je možné reverzní proxy použít pro zvýšení stability / dostupnosti
    - Proxy pak může fungovat ve dvou režimech a rozdělovat provoz mezi dva a více zdrojů
      - Fail-over
        - » Dostupných zdrojů mám sice více, ale používám jen jeden a další přepnu až v případě výpadku primárního zdroje
      - Balancer
        - » Má více zdrojů, které používám současně a rozděluji mezi ně provoz
        - » Krom toho, že zvyšuji dostupnost jako fail-over, dovedu zde i provoz škálovat do šířky
        - » Dnes typická konfigurace většiny větších serverů

# L7 - Aplikační vrstva: Obecně známé a používané protokoly

- Aplikačních protokolů je dnes obrovské množství
- Ale základních a nejčastějších byste měli vědět
  - K čemu slouží a jak cca fungují
  - Na jakých portech a jakými protokoly fungují
- Jedná se především o :
  - HTTP(TCP/80) / HTTPS(TCP/443)
  - POP3(TCP/110) / POP3S(TCP/995) / IMAP( TCP/143/220 ) / IMAPS( TCP/993 )
  - SMTP( TCP/25 ) / SMTPS ( TCP/587 )
  - TFTP( UDP/69 ) / FTP( TCP/20+21 ) / SCP( TCP/22 ) / RSYNC (TCP/873/22)
  - Telnet ( TCP/23 ) / SSH( TCP/22 ) / RDP( TCP/3389 ) / VNC ( TCP/5900)
  - DHCP ( UDP/67+68 ) / DNS ( TCP+UDP/53 )