Jak reprezentovat zprávy? **Formát dat.**Jak přenést zprávy po médiu? **Kódování dat.**Jak zajistit správnost přenosu? **Detekce chyb.**Jak identifikovat cílové zařízení? **Adresování.**Jak nalézt cestu k cílovému zařízení? **Směrování.**Jak dostat data na cílové zařízení? **Přepínání.**

Jak zajistit spolehlivý přenos dat?  
Jak efektivně využívat dostupných zdrojů?  
Jak řídit komunikaci mezi aplikacemi?

Síťové aplikace nechtějí řešit problémy výše. Pro aplikace chceme jen nějaké „virtuální komunikační kanály“, které jim poskytuje operační systém.

Zásadním cílem je umožnit aplikacím **spolehlivou** komunikaci. Na *IP vrstvě* (síťová, internetová vrstva, viz dále…) komunikace není spolehlivá, nastává:

* Ztráta dat
* Poškození dat
* Zpoždění
* Změna pořadí zpráv

Internet (= OS…) se to na *transportní vrstvě* (pod *aplikační vrstvou*) snaží kompenzovat! Různé mechanismy, jak to řešit efektivně – bude na to přednáška.

# Připojení k ISP

Různé typy připojení: sítě pro přístup domácností, institucí, mobilní přístup…

Různá kritéria: šířka pásma, agregace, cena.

## Domácí sítě

SOHO router – „*small office/home office router*“ – často směrovač spolu s modemem, firewallem, NATem, bezdrátovým přístupovým bodem, Ethernetem.

V minulosti: připojování pomocí telekomunikačních sítí. Do telefonní přípojky se připojil **modem**. Domácí modem vytočil speciální telefonní číslo, na straně ISP seděl další de facto modem, který si pak povídal s těmi domácími. Maximální rychlost: **56 Kbps**.

Připojování pomocí infrastruktury kabelové televize: **kabelové modemy**. HFC – hybrid fiber coax. Často asymetrické, ale vyšší rychlosti: cca 30 Mbps download, 2 Mbps upload. Obrovská agregace – sdílení linky minimálně se zbytkem paneláku… nebo čtvrti – problém zahlcení, dimenzování.

### ADSL

Lepší přístup k přenosu po telekomunikačních sítích. Spodní část frekvenčního spektra (do 4 kHz) se vyhradila pro hovory, vyšší část (od 25 do cca 1 100 kHz) se rozdělila pro upstream a downstream internetovou komunikaci.

Postupně vznikaly inovované varianty – používají jiný způsob reprezentace dat, a tak dosahují vyšších rychlostí. Problém: vzdálenost od ústředny.

### Bezdrátová připojení

Obvykle na volných pásmech – hodně stanic = větší rušení, nižší rychlosti.

### Mobilní sítě

Využívají rezervovaná pásma.

GPRS/EDGE (2G) -> WCDMA/HSPA/TD-SCDMA (3G) -> LTE, LTE Advanced (4G)

5G: nová licenční pásma (ukrojená z televizního vysílání); využívá menších buněk a menších antén, které směrují signál k zařízení

Linková (L2) a fyzická (L1) úroveň spolu obvykle souvisí (Ethernet, Wi-Fi, optika…)

Síťová vrstva (L3) – už společná pro internet – společný protokol IP (v4, v6); IP adresy říkají, co mají s paketem dělat

Transportní vrstva (L4) – UDP nebo TCP segmenty

Aplikační vrstva (L5) – protokol už závisí na konkrétní aplikaci

Na každé vrstvě se k bloku dat přidávají nějaké hlavičky

Klient-server / Peer-to-peer architektura (aplikace je „zároveň klientem i serverem“)

## Klient-server aplikace

Parametry:

* Je tolerovatelná ztráta dat?
* Propustnost (rychlost přenosu)
* Citlivost na zpoždění

TCP:

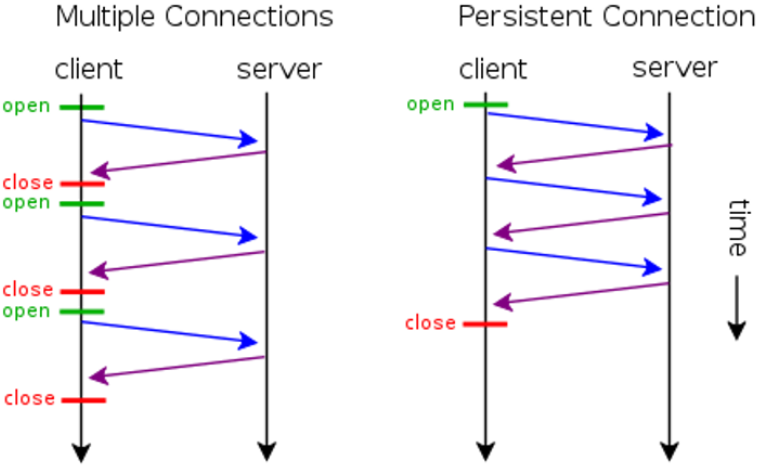
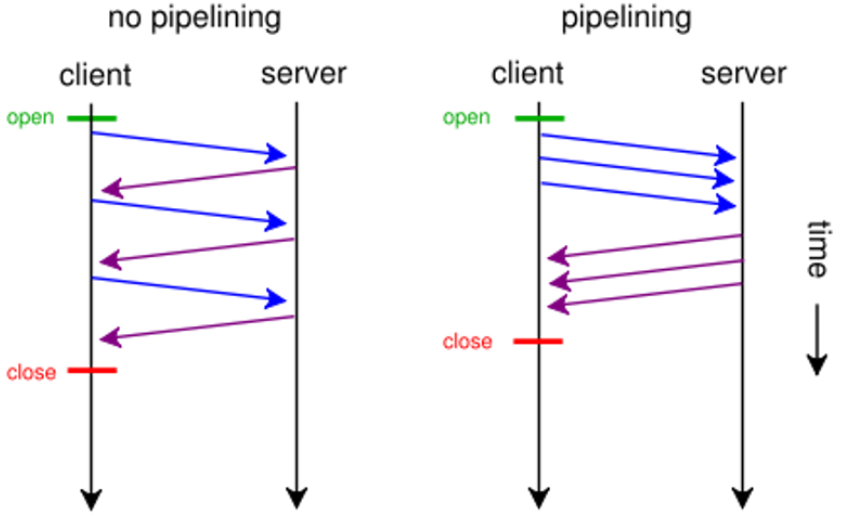
* Spolehlivá komunikace, vyšší režie
* Odesílatel je informován o ztracených paketech a musí jej poslat znovu
* Řeší správné pořadí odeslaných a přijímaných paketů
* Poskytuje mechanismus pro optimalizaci rychlosti přenosu dat tak, aby se jich co nejméně ztrácelo

UDP:

* U aplikací, kde není spolehlivost úplně potřebná
* Neřeší ztracené pakety, neřeší pořadí příjmu
* V hlavičce prakticky jen source/destination port, délka a checksum

# HTTP a web

Bezstavový protokol – server si nesnaží zapamatovat, co jeden konkrétní člověk dělal předtím.  
Na nižší vrstvě využívá TCP – první malá zpráva s požadavkem na spojení (nějaká režie), pak až samotná HTTP zpráva (=> 2x *round trip time*). Dřív to nevadilo, nyní se často používá možnost persistentního spojení a pipeline režimu (TCP umožňuje „najednou“ komunikovat oba směry).

Prohlížeče to různě kombinují – někdy si otevírají více TCP spojení najednou a požadavky dělají paralelně (to už je mimo HTTP – jednotlivé relace HTTP o sobě neví).

Řádky ukončeny vždy pomocí CRLF!

První řádek odpovědi – status line.

# Transportní vrstva (L4)

Má za cíl vzít data proudící z/do aplikací, nakrájet je na segmenty (**segmentation**), pustit je do cíle, tam je přijmout a slepit zpátky (**reassembling**).

Protokoly: **TCP**, **UDP** a v poslední době občas další, které se snaží najít něco mezi.

Zařizuje také multiplexing mezi jednotlivými *aplikacemi*.

Rozhraní mezi aplikací a transportní vrstvou = **socket**.

## UDP

[RFC 768](https://tools.ietf.org/html/rfc768).  
Koncept *best-effort*, nad kterým funguje IP (L3). Velmi jednoduchý protokol, který nic negarantuje.  
**Ne**ustavuje spojení – **connection-less** – neprobíhá žádná domluva na parametrech spojení (**nemá handshake**).

Typicky vhodný pro aplikace, jejichž protokoly fungují na jednoduchém principu request-response-nazdar, multimediální aplikace.

### Datagram

0 7 8 15 16 23 24 31

+--------+--------+--------+--------+

| Source | Destination |

| Port | Port |

+--------+--------+--------+--------+

| | |

| Length | Checksum |

+--------+--------+--------+--------+

|

| data octets ...

+---------------- ...

Dnes checksumování na úrovni L4 nedává smysl, protože to zařizuje nižší vrstva.

### Porty

K odeslání dat je třeba znát číslo portu aplikace. IANA udržuje katalog rezervovaných portových čísel:  
well-known ports 0–1023  
registered ports 1024–49151 (za ty si můžu zaplatit).

## Spolehlivý přenos

Specifikace UDP má 3 stránky, TCP má v základu 80 stran + spoustu rozšíření s dalšími rozšířeními.

Spousta služeb závisí na spolehlivé komunikaci, ale transportní vrstva využívá nespolehlivých přenosových kanálů.

Hypotetický spolehlivý protokol – RDT – lze popsat konečným automatem. Viz slidy. Je spolehlivý, ale má *tragickou utilizaci* – velmi málo využívá linku.

Řešení: zřetězené protokoly (**pipelining**) – vždy chci udržet co nejvíce balíků na cestě.  
Dva přístupy:  
**go-back-N** (příjemce odesílá kumulativní potvrzení, nejstarší nepotvrzený paket má časovač, když vyprší, posílá se znovu celé okno)  
**selective repeat** (příjemce odesílá individuální potvrzení, zavádí potřebu bufferu u odesílatele i příjemce).

Mechanismy: kontrolní součty, časovače, sekvenční čísla, potvrzení (pozitivní/negativní), okna, zřetězení.

## TCP

Spolehlivé doručování v pořadí. Řízení toku (**flow control**), zahlcení (**congestion control**). Ustanovuje spojení.

Pro aplikace to je *blackbox* – na úrovni L4, síťového stacku, OS se zajišťuje, že ty zprávy dojdou tak, jak mají, aplikace o tom vlastně neví.

Vždy je point-to-point – právě jeden odesílatel, právě jeden příjemce.  
Garantuje zachování pořadí **bytů** – *in-order byte stream* – bez ohledu na obsah se krájí po bytech, nezachovává hrany mezi aplikačními zprávami.  
Používá koncept zřetězeného přenosu – parametr N = window size zajišťuje řízení toku a obranu vůči zahlcení.  
Využívá vyrovnávacích pamětí pro odesílaná i přijímaná data.  
**Spojově orientovaný** – connection-oriented – mezi stranami se drží otevřené spojení, které se explicitně zahajuje a ukončuje nějakou sekvencí zpráv.

0 10 20 30 |

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Port | Destination Port |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Acknowledgment Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data | |U|A|P|R|S|F| |

| Offset| Reserved |R|C|S|S|Y|I| Window |

| | |G|K|H|T|N|N| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Checksum | Urgent Pointer |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

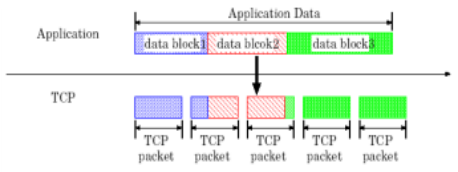
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

### In-order byte stream

Data přenášená TCP segmentem nemají žádnou strukturu, jde o proud bytů.



### Sekvenční a potvrzovací čísla

SEQ – číslo prvního bytu odesílaného segmentu  
ACK – číslo prvního bytu **dalšího očekávaného** segmentu k přijetí

V rámci jedné zprávy zároveň odesílá vlastní data a potvrzuje přijetí dat z druhé strany.

### Úvodní parametry spojení

Initial Sequence Number – počáteční SEQ číslo.  
Maximum Segment Lifetime – jak dlouho může balík sítí putovat (= jak dlouho má smysl něco držet v bufferu).  
Maximum Segment Size – maximální délka balíku. Snažíme se ji přizpůsobovat podle nižších vrstev (síťová, linková).

### Datový tok

**Flow** = zdrojová IP+port, cílová IP+port, protokol (TCP).

#### Zahájení spojení:

Active-active (velmi výjimečně), obvykle active-passive.  
Využívá se 3-way handshake (2-way může vést na napůl otevřené spojení nebo na vkládání dat):  
SYN (klient) -> SYN+ACK (server) -> ACK (klient) -> spojení navázáno

#### Ukončení spojení:

Jedna ze stran nastaví bit FIN, čímž říká, že už nemá žádná data k odeslání, pořád ale musí přijímat!

K znovu zasálání zpráv dochází když vyprší časovač nebo když se obdrží duplicitní ACK.  
**Kumulativní potvrzování**: když mi nedojde ACK na balík ukončený bajtem #100, ale pak dojde ACK na balík s bajtem #120, vím, že nemusím znovu posílat ten první balík. S tím souvisí **retransmission** (slidy).

### Nastavení timeoutu

Timeout potvrzení TCP segmentu by měl být vyšší než je RTT (round trip time). Ten ale musím nějak odhadnout a hlavně se může měnit.

*EWMA + SRTT*

### Fast retransmit

Ztracené segmenty lze poznat pomocí duplikovaných ACKs – jakmile několikrát přijde (obvykle 3x), nečekám na timeout a rovnou to pošlu.

Lepší možnost: **selektivní ACK**.

### Příznak URG

Jednoduchý mechanismus pro specifikaci prioritních dat – URG příznak.

### Příznak PSH

= push. Pokud jej odesílatel pošle příjemci, příjemce by měl okamžitě všechno vystrčit z bufferu do aplikace.

### Řízení zahlcení

Zahlcení nastává, když je množství dat k přenesení vyšší než přenosová kapacita linky.  
Všechny kapacity jsou konečné, omezené.  
Zahlcení má tendenci se zvyšovat, zhoršovat – protokoly opakovaně znovuzasílají data, uživatelé se donekonečna pokouší o přenos, což generuje ještě větší provoz.

Přístupy k řízení zahlcení: **end-end** (síťová vrstva neposkytuje žádnou podporu, koncové stanice sledují, jak se síť chová – TCP, internet celkově; routery a switche mohou být vcelku hloupoučké) a *network-assisted* (komponenty síťové vrstvy poskytují informace o stavu sítě, speciální choke packet).

Základní princip: pokud se pakety neztrácejí, zvyšuji rychlost přenosu – zvětšuji okno (kolik bajtů se najednou na síti vyskytuje). Pokud se ztrácejí, snižuji, zmenšuji.

**Congestion window *cwnd***

AIMD – additive increase, multiplicative decrease – zvyšování rychlosti je lineární, snižování je geometrické.

# Síťová vrstva (L3)

Stará se o přeposílání packetů z jednoho rozhraní na druhé.  
Klíčovým prvkem je **směrovač – router.** Rozhodování, kterou cestou (**route**) bude packet vyslán = **routing**.

**Connection-less**: přístup k budování sítí spočívá v bezokruhovosti, síť je jednoduchá a o komunikaci v ní se starají zejména koncová zařízení, která jsou celkem chytrá. Většina sítě je bezstavová, jediným stavovým prvkem jsou **směrovací tabulky** v routerech; nic víc si routery nepamatují. Síť je prostě otrokem, který přehazuje packety z jedné strany na druhou. **Best effort delivery** – snaží se doručit, ale není to zajištěno.

**Connection-oriented**: mezi jednotlivými prvky se ustavuje spojení – virtuální **okruh**. Ta síť musí být dostatečně chytrá. Když už se okruh vyjedná, má nějakou trvalou charakteristiku (může mít například vyhrazenou šířku pásma). Síť je schopná aktivně řešit ztrátu paketu, pořadí, zpoždění atp. Krabičky na koncích jsou vcelku hloupé a dovolují používat jen služby, které jim dovolí ta síť.

ISO/OSI stack je velký kočkopes vzniklý diskusí mezi CL a CO přístupem.

**Záznam v routovací tabulce**: Zdroj – Adresa + Maska – Administrativní distance + Metrika (nižší = lepší) – Next hop adresa – Outgoing interface.

Zdroj: **C** (přímo připojeno na rozhraní) / **D** (nějakou chytristikou zjištěná) / **S** (statická, adminem definovaná).  
Next hop se u přímo připojených neuvádí.

## IPv4

RFC 791, rok 1981.

Designové rozhodnutí – pokud má počítač víc rozhraní, každé má vlastní adresu, jsou to adresy rozhraní, ne počítače. Moc špatné.

Původní RFC jen velmi, velmi hrubě naznačuje nějakou podporu pro typy služeb a zajišťování Quality of Service (Type of Service políčko v hlavičce), ale adoptovat tu myšlenku jim trvalo asi 17 let.

Základní velikost hlavičky je **20 B** (+ options v TLV formátu – type, length, value, moc se nepoužívají).

IHL: Internet Header Length – velikost IP hlavičky. Může být různá, protože na konci hlavičky jsou nějaké extra options.

Type of Service: dva typy – integrované / diferenciované služby (v ToS políčku je nějaký flag o typu provozu).

Příznaky DF – don’t fragment, MF – more fragments, fragment offset.

**TTL**: Time to Live – dřív v něm bývalo časové razítko, maximum packet lifetime. Hodiny ale nejsou napříč světem synchronizované, takže se tam začal strkat counter dekrementující se při každém hopu.

Header Checksum: vůči Ethernetu (L2) jsou redundantní.

### Maximum Transmission Unit

Linková vrstva má nějakou maximální velikost balíku, kterou je ochotna poslat. (Linka musí spravedlivě rozdělovat šířku pásma mezi své účastníky.) U basic 10 Mbps Ethernetu je to 1500.

Když je MTU linky menší než velikost balíku na L3, začne se fragmentovat – balík se nakrájí, každý kousek se označí vlastní hlavičkou jako fragment (MF bit, fragment offset) a odešle. IP datagram zpátky sestavuje až koncový prvek.

## Adresování

IPv4 používá adresu pevné délky – 32 bitů – zapisujeme jako decimální hodnoty čtyř bajtů oddělené tečkou.

Adres je 232-2 (samé nuly a samé jedničky jsou rezervované).

Každá adresa se skládá ze 2 částí: identifikace sítě (**NetId**) + identifikace hosta (**HostId**). Potřebujeme to vědět, abychom byli schopni určit, jestli má packet někam směrovat router, nebo jestli se doručuje napřímo (resp. přes switch). Otázka: jak určit hranici mezi nimi?

Původní koncept: třídní adresování – classful addressing – třídy adres A–E, které říkají, kde ta hranice je.

Současnost: beztřídní adresování – classless addressing – adresa obsahuje navíc masku (**VLSM**).

**Bez masky nedává adresa smysl!**

Typy adres:  
Unicast (v prostoru A, B, C): konkrétní počítač  
Broadcast (same): všechny počítače v jedné síti  
Multicast (v prostoru D)

* { network , 0 } – “This” Network
* { network , -1 } – Directed Broadcast
* { -1 , -1 } – Limited Broadcast, pouze jako cílová adresa
* { 0 , 0 } – All Zeros IP Address, pouze jako zdrojová adresa
* { 0 , host } – IP address on This Network
* { 127 , <any> } – Software Loopback

**CIDR** – Classless Interdomain Routing.

### Adresní prostor

Nejvyšší organizace: **IANA** – stará se „o všechna čísla v TCP/IP stacku“ – nejen různé konstanty, ale i adresy. V podstatě jen udržuje jeden obří katalog. Později vznikla exekutivní organizace **ICANN**, ve kterém jsou zástupci většiny států (*nějaká paní z ministerstva vnitra*). Pod tím jsou **RIR** – „regionální internetový registrátor“ (kontinentální…) – u nás je to **RIPE NCC**. Zákazníky RIRů jsou **LIR**ové – lokální internetoví registrátoři.

RFC 1597 definuje privátní adresy, které jsou v internetu neroutovatelné (v bloku A 10.0.0.0/8, v bloku B 172.16.0.0/12, v bloku C 192.168.0.0/16).

## ICMP

Internet Control Message Protocol – používán hosty a směrovači pro diagnostiku sítě a oznamování chyb.

Dva typy zpráv: oznámení chyby, dotaz na službu. Dál viz slidy.

## DHCP

Dynamic Host Configuration Protocol.  
Rozšíření protokolu BOOTP – historická záležitost.  
UDP porty 67 (destination port) a 68 (source port).

První zpráva – DHCP discover. Počítač naprosto netuší, kde je, proto do té zprávy strčí svou MAC adresu, jako source IP address uvede 0.0.0.0 a posílá to na všesměrový broadcast – samé jedničky (v dst IP i dst MAC).

Odpověď – DHCP offer. Už tady se objevuje klientova IP adresa jako destination v IP hlavičce.

Odpověď klienta – DHCP request. Klient říká „OK, beru ty parametry, co jsi mi dal“. Pořád je broadcastová**!**

Odpověď serveru – DHCP acknowledgement.

IP adresa je vypůjčena na nějakou dobu – IP lease. Pro obnovení se typicky pošle jen DHCP request.

### Mimo subnet

V rozsáhlých sítích chci mít typicky jeden DHCP server někde na jednom místě, ale DHCP requesty se posílají na broadcast, který končí routerem. K tomu se používá **DHCP relaying** – na routeru je nakonfigurováno, že všechny DHCP packety vezme, přebalí do unicastu a pošle směrem na ten server. Taky to využívá rozšíření DHCP „**Option-82**“.

## NAT

Přepisování IP adres. Existuje statický (1:1) NAT, dynamický (N:M) NAT, ty přepisují pouze IP adresy (L3 IP hlavičky).

Typicky se ale používá overloading NAT/**port address translation (PAT)**, který přepisuje také portová čísla (sahá i do L4 hlaviček – TCP/UDP). Problém je u aplikací, které nějak mluví o portu i na aplikační vrstvě.

## Routing

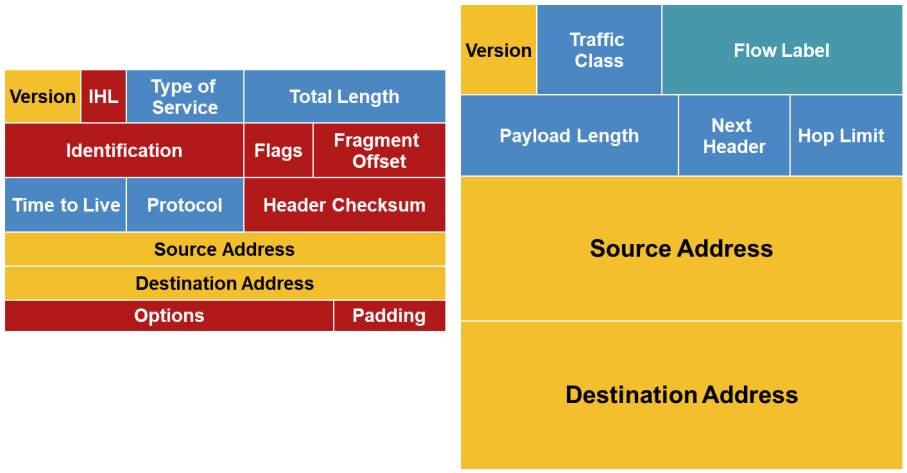
Doplnit ☹

## IPv6 (síťová vrstva)

V 90. letech se do výběru kandidátů pro náhradu IPv4 dostaly asi tři technologie.  
**Motivace**: větší adresní prostor, end-to-end princip, bezstavová autokonfigurace, zjednodušení hlaviček, bezpečnost, podpora kvality služeb (diferencované, integrované služby), lepší podpora multicastu a anycastu (haha, u wish), vylepšit mobilitu zařízení (opět haha – pořád označuje adresa rozhraní).

**Nové vlastnosti:**

* Jedno *rozhraní* má *více* než jednu IPv6 adresu.
* Fragmentace se děje *pouze* na koncových systémech, nikoliv na směrovačích – zvýšilo se minimální MTU (libovolná implementace musí být schopna odbavit 1280 B v jednom balíku) + Path MTU Discovery (pravidelně si od destinací, se kterými komunikuji, zjišťuji omezení MTU po celé cestě – implementační peklo though).
* Kontrolu bitových chyb dělá nižší vrstva (v hlavičce není CRC).
* Pevná velikost hlavičky (40 B).
* Nepodporován přímý broadcast.



IPv4 hlavička vs. IPv6 hlavička

### Řetězení hlaviček

**Extension headers** se připojují za základní hlavičku datagramu (v základní hlavičce odlišeno polem *Next Header*). V minulosti to bylo nepříjemné (nedostatečná podpora na aktivních prvcích – vhodným přeskládáním hlaviček mohly pakety prosakovat firewally…).

Na papíře je to pěkné, ale ve finále se to na aktivních prvcích dělá složitěji – čtecí engine musí projít všechny hlavičky (u IPv4 se prostě podívá na prvních pár bajtů a přeskočí celou hlavičku včetně options).

### Fragmentace a jumbogramy

Snaha co nejvíce omezit – fragmentuje výhradně odesílatel (leč je to navrženo tak, aby to zvládly dělat i routery). Směrovače s nedostatečným MTU pakety zahazují a informují odesílatele.

Předmětem fragmentace může být všechno **za** *fragmentovací hlavičkou*.

V IPv6 jsou podporované i velké datagramy, pokud to umí linková vrstva.

### Adresování

Délka adresy je **128 bitů** (16 bytů).  
Způsob zápisu upravuje RFC 5952:

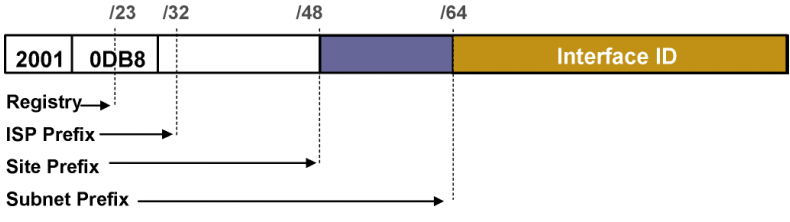
* Vždy malá písmena.
* Vynechávají se úvodní nuly (0db8 -> db8).
* Vynechává se nejdelší souvislá skupina nulových čtveřic – nahrazuje se dvěma dvojtečkami.
* 0000, které tam zbydou, se zkrátí na 0.

„Sítě, podsítě“ nahrazeny pojmem **linka**. IPv6 **uzel** má **více** **adres**. Adresy mají scope a lifetime (dosah a doba platnosti).

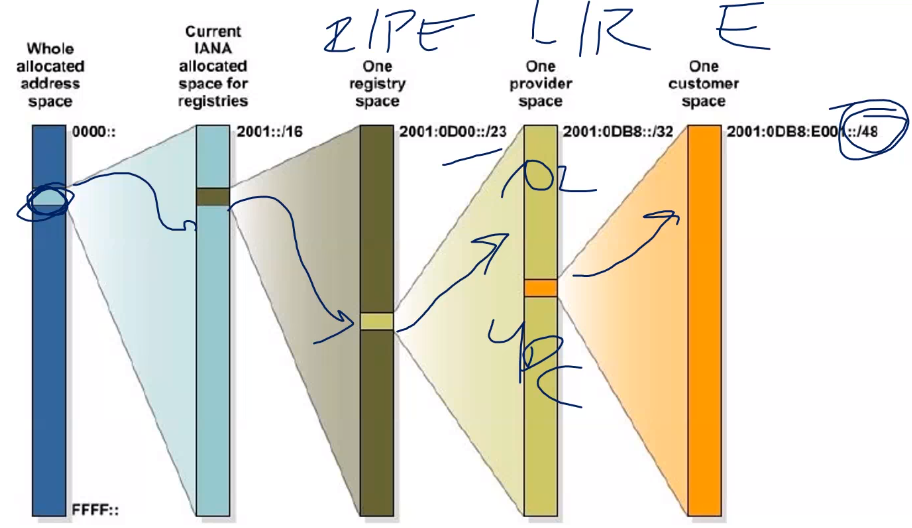
### Adresový prostor

Individuální (unicast) a skupinové (multicast) adresy.  
Součástí prostoru unicastu je anycast.

* Nespecifikovaná adresa – **::/128** (samé nuly, délka *prefixu* 128)
* Loopback (paralela k 127.0.0.1) – **::1/128**
* Link-local – **fe80::/10**
* Multicast – **ff00::/8**
* IPv4 mapované adresy – **::ffff:XXYY:ZZWW**
* Unique local (analogie k privátním adresám typu 192.168.) – **fc00::/7**



**Interface ID** („host ID“) je typicky /64 – v jednom „firemním subnetu“ je 264 adres!



Jak určit interface ID pro zařízení? Různé možnosti:

* Stabilní: EUI-64 – podle MAC adresy – typicky je používají routery.  
  Problém – zařízení by se dalo sledovat napříč internetem.
* Privacy Extension: hodnota úplně náhodná.
* Kryptograficky generované adresy.

### Stavy adresy

**Tentative**: zařízení si namyslelo nějakou adresu, ale nevíme, jestli už neexistuje. Unicast je zakázaný, z multicastu je povolen *neighbour advertisement*.

**Valid**: adresa je unikátní a může být používaná.  
*Preferred* Valid: adresa je používaná.  
*Deprecated* Valid: adresa je používaná v rámci stávajících spojení, ale žádná nová nesmí být vytvářena.

**Invalid**: není platná, nedá se používat.

### Link-Local adresy

Prefix **ffe80::/10** – 54 bitů je náhodných (obvykle nula), dalších 64 určeno zařízením. První adresa, která je vytvořena.

### Unique-Local adresy

Prefix **fc00** nebo **fd00::/8**.

### Solicited-Node adresy

Používají se pro komunikaci pomocí multicastového *neighbour discovery* protokolu!

Doplnit.

### Anycast

Vlastnost, kde několik zařízení má stejnou adresu a je na struktuře sítě, kam se paket doručí. Typicky například DNS kořenové servery.

**Motivace**: rozkládání zátěže – chytrým adresováním by mělo být dosažitelné, že budou data vždy doručena na nejbližší možné zařízení. Zmenšuje počet adres, na nichž je poskytována jedna *služba*.

**Problémy**: anycastová adresa se nedá poznat od unicastové (nemůžu se k tomu speciálně chovat). Jakási stavovost?

## Podpůrné protokoly IPv6 – ICMPv6

Vzalo funkcionalitu všech starších podpůrných protokolů – ICMP, IGMP, ARP (drobný detail: ARP je L2 protokol, ale při použití IPv6 se to dělá až na L3 – aplikuje se na něj třeba L2 firewall), DHCP…

Součásti:

* MLD
* ND
* …

### Objevování sousedů – neighbour discovery – ND

Řeší spoustu provozních otázek.

Router Advertisment (RA) – periodické zprávy směrovače oznamující jeho existenci, specifické paramerty jako prefix linky, MTU; vysílány také jako odpovědi na SA.

Router Solicitation (RS) – požadavek hosta na zaslání RA.

Neighbor Solicitation (NS) – požadavek na získání linkové adresy, pro detekci duplikované adresy

Neighbor Advertisment (NA) – odpověď na NS. Při změně linkové adresy vyšle unsolicited NA.

# Směrování (L3)

Probíhá na **síťové vrstvě** (umožňuje konektivitmu mezi zařízeními). Směrování je určování vhodné cesty (vhodné sekvence směrovačů) od zdroje A k cíli B.

Směrovače v síti potřebují vědět, jaký směrovač X použít pro dosažení cíle a jaké rozhraní použít, aby s tím směrovačem X mohly komunikovat.

Když ho zapojíme do zásuvky, neví o síti **nic** – potřebuje si síť nějak zmapovat – různé přístupy (celou síť projít najednou, nebo se postupně doptávat sousedů, co ví).

Směrovač *má* v tabulce informaci o správném rozhraní pro *každou* IP adresu. Kdyby to tak bylo, měla by tabulka 4 GiB – not great. Naštěstí jsme schopni celou síť (potenciálně mnoho počítačů) identifikovat nějakým NetID – síťovým prefixem (IP + maska).

Princip CIDR – classless interdomain routing – pro směrování můžeme agregovat síťové prefixy – do vyšších úrovní sítě můžeme propagovat nějakou obecnější informaci („zvládnu obsloužit 4 subnety /20, které pokrývají nějakou masku /18…“)

Pokud ve směrovací tabulce vyhovuje cílové adrese více položek, použije se ta, která má **nejdelší** prefix sítě – je „nejkonkrétnější“.

Vyhledávání je potřeba dělat efektivně (ne lineárně) – používají se např. **prefixové stromy**. Každá úroveň stromu představuje jeden bit v adrese. Některé uzly mají pak přiřazenou hodnotu. Nejideálnější je ale použít na to **speciální hardware**, který implementuje **asociativní pole**.

Na nejvyšších úrovních se mohou používat i nějaké speciální koncepty, které směrují jinak než přímo podle cílové IP adresy (*např. MPLS*).

#### Kde se berou položky v RT?

Položky mohou být přidány **staticky** (administrátorem) – nereflektují tak nutně aktuální stav. Typicky se tedy zjišťují pomocí **dynamických směrovacích protokolů** – IGP, EGP.

#### Z pohledu koncové stanice

Pokud má jen *jedno* rozhraní, potřebuje znát jen vlastní IP adresu, IP adresu brány a síťový prefix (masku).

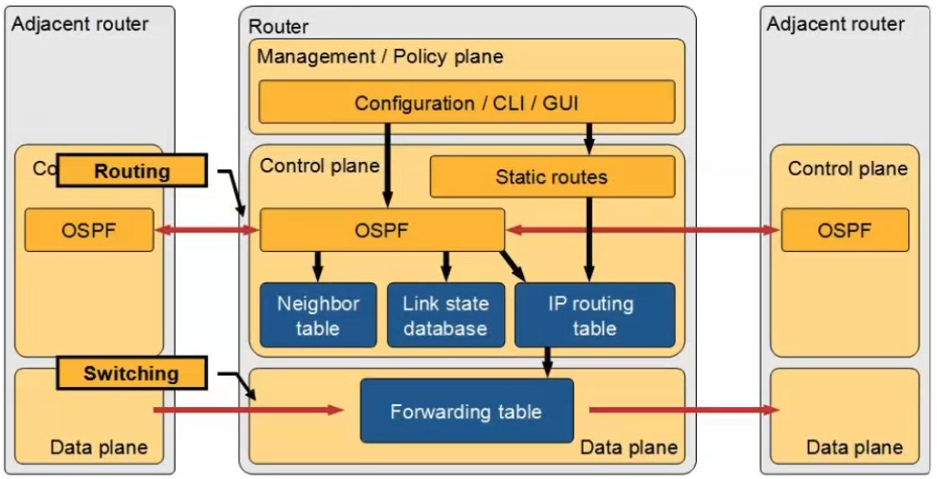
Paket na vlastní je doručen lokálně, paket ostatním uzlům ve stejné síti se posílá přímo v ethernetovém rámci (zeptá se pomocí ARP, čí ta IP adresa je) a paket do vnější sítě je poslán na bránu.

Typicky ale mají i koncové stanice směrovací tabulky (protože mívají více interfaců) – „jediný rozdíl“ od směrovače je v tom, že typicky zahazuje pakety, které nejsou pro ni, nesnaží se je směrovat dál.

## Architektury směrovačů

Jsou to prostě počítače.  
Obvykle mají nějakou implementaci RT, směrovacích protokolů a přepínání paketů.

Další funkce: viz prezentace.



**Data plane**: zpracovává pakety – načte paket, podívá se do tabulky, pošle paket dál. Prioritou je **rychlost**.

**Control plane**: tam běží ty směrovací protokoly, vyplňuje forwarding (routing) table.

**Management plane**: poskytuje rozhraní pro správu konfigurace.

### Sběrnice s centrálním CPU

Pomalé.

### Sběrnice s lokální cache

Na jednotlivých síťových kartách jsou částečné kopie centrální směrovací tablky, mezi kartami stále sběrnice. Pakety, které karta nezvládne zpracovat přímo, jdou dál do centrálního CPU.

### Sběrnice s paralelním zpracováním

Viz prezentace.

## Směrovací algoritmy

### Bellman-Fordův algoritmus

Používá se u **distance-vector** protokolů. Na jednotlivé linky jsou nějak přiřazeny *ceny* – cílem je získat cestu s nejnižší cenou.  
Na začátku směrovač nic neví, komunikuje se sousedy, aby zjistil více – *routing by rumour*.  
Směrovače aktivně svým sousedům posílají nové informace.

Je tedy **iterativní, asynchronní**.

Funkce: viz přednáška 10, 01:33.

# Linková a fyzická vrstva (L2, L1)

L2 – to, co umožňuje komunikaci mezi dvěma zařízeními. Pomocí těchto prostředků komunikujeme **uvnitř jedné sítě**, „na jednom segmentu“ (tím mohou být i dva propojené směrovače).

Doručuje **rámce** linkové vrstvy mezi **sousedními počítači v lokální síti** – na *jedné lince*.  
**Rámec** L2 zapouzdřuje pakety síťové vrstvy. I na L2 řešíme problém **adresování**.

Existují různé technologie „LAN“, můžeme je různě klasifikovat (podle řízení přístupu, podle řízení přenosového média – kabel vs. vzduch, half-duplex vs. full-duplex).

Technologie řízení linkové vrstvy bývá typicky implementována **v HW** – v **síťovém adaptéru** – NIC – network interface card. V SW je realizováno vytváření rámců, adresování.

Služby L2:

* **Přístup k médiu** (zabalení dat do rámců, přístup ke sdílenému médiu, adresování),
* **Spolehlivé doručení** (záleží na technologii, Ethernet ne)
* **Detekce a oprava chyb**

Detekce chyb: viz INP – např. parita a další kódy (klasifikujeme podle schopnosti detekce n-násobných chyb a opravy m-násobných chyb – Hammingova vzdálenost).

### Vsuvka: IPv4 checksum

Data jsou sdružena do 16b slov, ta jsou sečtena a udělá se jedničkový doplněk (otočí se bity), který se uloží do checksum pole. Při kontrole je vypočtena suma všech polí, pokud je výsledek 0xFFFF, je to ok.

### CRC kódy

Podle teorie polynomiálních kódů. Nějakým generátorem se vygeneruje nějaký polynom, tím se zpráva vydělí a vyšlý zbytek je kontrolní kód. Dá se to snadno udělat v HW (posuny, XOR).

## Typy linek – topologie

Dvě základní kategorie: **bod-bod** (point-to-point) – tím je třeba moderní přepínaný Ethernet s přepínačem mezi dvěma stanicemi; nebo **multi-access**.

### Řízení multi-access přístupu

Máme jedno sdílené médium, na kterém se snaží více uzlů vysílat v jednom čase – vzniká interference. Chceme férově rozdělovat dostupnou kapacitu a zajistit, že pouze jeden uzel může úspšně přenést zprávu.

Potřebujeme **protokol pro řízení vícenásobného přístupu** – distribuovaný algoritmus, který určuje, jak budou uzly ten kanál sdílet, kdy mohou vysílat data.

Ideální protokol by měl být **decentralizovaný**, umožňovat uzlu vysílat **plnou rychlostí**, rozdělovat komunikaci **férově – rovnoměrně**, neměl by vyžadovat synchronizaci hodin a měl by být jednoduchý.

Druhy protokolů:

* Rozdělující přenosový kanál
* S náhodným přístupem – dovolí kolize, ale potom se z nich zotaví
* S koordinovaným přístupem – přísný, vyzvedávání a předávání *žetonu*

## Protokoly rozdělující přenosový kanál

### TDMA – Time Division Multiple Access

Jsou definovány časové **intervaly**, kdy může stanice k médiu přistupovat; intervaly se střídají v rámci **kola**. Nevyužité sloty jsou prázdné – při nižším využití zbytečné zpoždění.

### FDMA – Frequency Division Multiple Access

Dostupné pásmo je rozděleno do oblastí podle **frekvencí** (-> Fourierovo peklo). Opět je „spravedlivé“, ale když ho někdo nevyužívá, zbytečně leží ladem.

### CDMA – Code Division Multiple Access

Všichni sdílejí stejnou frekvenci i čas, ale každý uživatel má přiřazen **jedinečný** identifikátor – **kód**. Tím se zprávy kódují. Používá se často u bezdrátových sítí.

Podmínka ortogonality: pokud kódy pronásobíme po složkách a sečteme, vyjde nula. (viz přednáška – je to prostě lineární algebra).

## Protokoly s náhodným přístupem

### Slotted ALOHA

Všechny rámce mají stejnou velikost, čas je rozdělen do stejných slotů (potřebných pro přenos jednoho rámce) – uzly mohou začít vysílat pouze na začátku slotu -> uzly musí být **synchronizované** (což úplně nechceme). Všechny uzly detekují kolize.

Výhody: „decentralizované“ (mimo synchro), jednoduché, je možné využít plné kapacity média.  
Nevýhody: kolize plýtvají sloty, kolize je možné detekovat dříve.

Dlouhodobá efektivita maximálně cca **37 %** – velké plýtvání! **Výpočet viz slidy**.

### Pure ALOHA

Bez synchronizace, komunikace začíná kdykoliv – bez čekání na nějaký slot. Vzrůstá pravděpodobnost kolize – je to ještě horší – maximální využití je nějakých **18 %**.

### CSMA – Carrier Sense Multiple Access

První se podívám, jestli je médium zrovna volné. Pokud není, odešli rámec, jinak čekej nějakou +- náhodnou dobu.

Bohužel to neodstraňuje všechny kolize – v tom médiu nějakou dobu trvá šíření toho signálu.

### CSMA/CD – Collision Detection

Po detekci kolize je přenos okamžitě ukončen – šetří přenosovým pásmem. Back-off algoritmus: když je detekováno více kolizí, další pokus o komunikaci se odloží o něco dál.

### Polling protocol

Máme nějakou centrální stanici, která vyzývá ostatní k přenosům. Má dost velkou režii a selže, když selže master uzel.

# Ethernet

IEEE 802.3. Nejpoužívanější LAN technologie současnosti, velmi levný, snadný na správu a spolehlivý. Na začátku sdílené médium s half-duplex přenosem, nyní se používá obvykle jako P2P s full-duplex.

Rámec:

* 8 B preambule
* 6 B dest address
* 6 B source address
* 2 B typ
* 46–1500 B: data
* 4 B: frame check sequence

**Nespojovaná** komunikace(každý rámec má zdroj a cíl, žádná synchronizace).  
**Nespolehlivá** komunikace (jen kontrolní součet).

Pro full-duplex bezkolizní, pro half-duplex CSMA/CD.  
V případě kolize vysílají strany 32b posloupnost **JAM**. Aby to fungovalo, musí mít rámce **minimálně 64 B** a musí být **omezena délka kabelu**.

Maximální počet kolizí najednou je **16** – jinak excessive collisions error.

Efektivita – výpočet viz slidy.

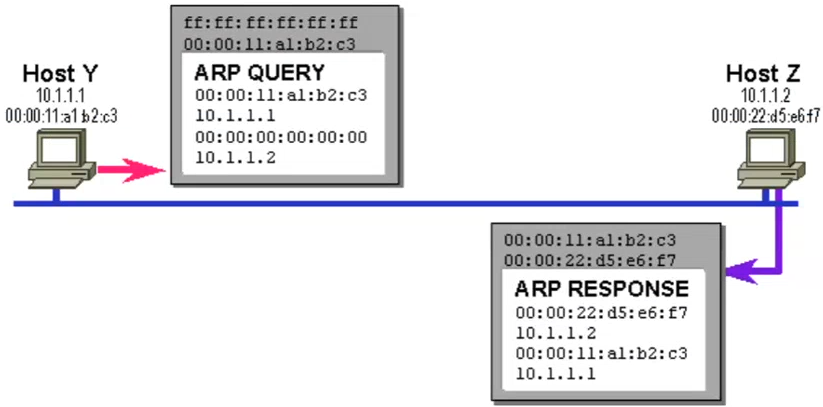
## MAC adresy

48bitová adresa typicky vypálená v **ROM** paměti adaptéru. Každý adaptér v lokální síti musí mít unikátní adresu. Rozděleno výrobcům organizací IEEE.

**Broadcastová adresa**: samé jedničky (FF-FF-FF-FF-FF-FF).  
**Multicastový prefix**: 01-00-5E a jedna fixní nula za tím – pro kódování multicastové IP adresy je použito spodních **23 bitů**.

### ARP – Address Resolution Protocol

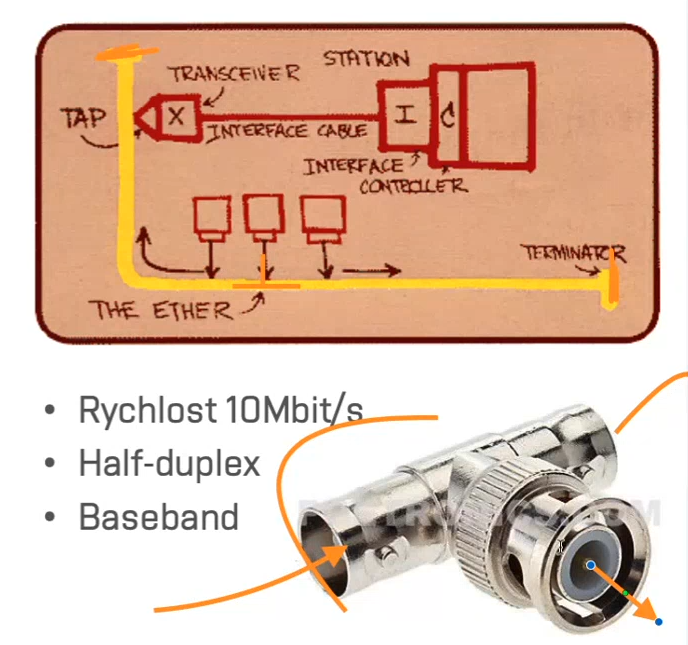
Každý IP host má ARP tabulku, která mapuje IP na MAC. Je to jednoduchý protokol:  
Stanice pošle ARP broadcast, všechny stanice přijmou ARP broadcast a jiná stanice, která má danou IP adresu, odpoví **unicastovou** odpovědí.



Při směrování mimo LAN (podle IP konfigurace) se odesílají ethernetové rámce s adresou **brány** – toho routeru, který nás do cílové sítě může dostat.

## OG ethernet

Na začátku se používaly koaxiální kabely – 10Base2 (tenký koax, 200 m), 10Base5 (tlustý koax, 500 m).



## Dnešní ethernet

„Fast Ethernet“ – kroucená dvojlinka – čtyři páry vodičů (ale používají se jen dva – Rx a Tx) – 100BaseT – diferenciální signály.

Data se přenáší pomocí **Manchester kódování**.

Gigabitový ethernet – používá všechny vodiče a kódování PAM-5 – pět napěťových úrovní.

## Zařízení linkové vrstvy

Dřív se používaly opakovače – **repeatery.** Dnes se ale používají téměř výhradně přepínače – **switche.** Zkoumají hlavičky rámců a selektivně přeposílají rámce podle **Content Address Memory** (CAM) na základě cílové adresy na příslušný port**.**

Pozor – routery, směrovače, jsou zařízení L3 – **rozdělují broadcastovou doménu**!